

IOI - International Olympiads in Informatics

- 1st IOI 1989 - Πράβετζ, Βουλγαρία (ok)
- 2nd IOI 1990 - Μίνσκ, Λευκορωσία (ok)
- 3rd IOI 1991 - Αθήνα, Ελλάδα
- 4th IOI 1992 - Βόννη, Γερμανία
- 5th IOI 1993 - Μεντόζα, Αργεντινή
- 6th IOI 1994 - Χάνινγκε, Σουηδία
- 7th IOI 1995 - Αϊντχόβεν, Ολλανδία
- 8th IOI 1996 - Βέτσπριμ, Ουγγαρία
- 9th IOI 1997 - Cape Town, South Africa

BOI - Balkan Olympiads in Informatics

- 1st BOI 1993 -
- 2nd BOI 1994 -
- 3rd BOI 1995 -
- 4th BOI 1996 -
- 5th BOI 1997 - Drama, Greece

All Ireland School Contest

USACO

The Following styles have shortcuts :

Ctrl-Shift-X Small Normal

Ctrl-Shift-A Para Title

Ctrl-Shift-E Exercise

Ctrl-Shift-N Normal

1η Διεθνής Ολυμπιάδα Πληροφορικής 1989	6
Πρόβλημα 1. Μεταθέσεις Κουτιών	6
Πρόβλημα 2. Ανελκυστήρες σε κτίριο	6
Πρόβλημα 3. Το βιβλίο που θα διαβάσουν όλοι	7
Πρόβλημα 4. Κωδικοποίηση για Επικοινωνίες	7
Πρόβλημα 5. Αριστερός και Δεξιός γείτονας	7
Πρόβλημα 6. Εικοσάεδρο	8
2η Διεθνής Ολυμπιάδα Πληροφορικής 1990	9
Γύρος 1ος	9
Πρόβλημα 1. Puzzle με αριθμούς	9
Πρόβλημα 2. Παιχνίδι με Κουκίδες	9
Πρόβλημα 3. Βιβλία για δύο αναγνώστες	10
Πρόβλημα 4. Παιχνίδι για δύο παίκτες με συνεχόμενα κελιά	10
Πρόβλημα 5. Γλώσσα PROLAN/M	11
Πρόβλημα 6. Υψωση σε δύναμη	12
Γύρος 2ος	12
Πρόβλημα 1. Φύλακες σε γκαλερί	12
Πρόβλημα 2. Ευθύγραμμα τμήματα και τέμνουσα ευθεία	12
Πρόβλημα 3. Η κίνηση των ρομπότ	12
Πρόβλημα 4. Ανηφόρες, κατηφόρες σε τετράγωνη πόλη	13
3η Διεθνής Ολυμπιάδα Πληροφορικής 1991	14
Πρόβλημα 1. Πασιέντσα με Τράπουλα	14
Πρόβλημα 2. Η περιφραξη των δέντρων	14
Πρόβλημα 3. Τετράγωνο (επιλέχθηκε για τον 1ο γύρο)	15
Πρόβλημα 4. Οι ξένες γλώσσες	15
Πρόβλημα 5. S-TERMS (επιλέχθηκε για τον 2ο γύρο)	16
Πρόβλημα 6. Η μέγιστη συμμορία	17
Πρόβλημα 7. Τα ραντεβού του Ιατρικού Επισκέπτη	17
4η Διεθνής Ολυμπιάδα Πληροφορικής 1992	19
Πρόβλημα 4.1.1. Μυστηριώδεις Ηπειροι	19
Πρόβλημα 4.1.2. Εργαστήριο λαβυρίνθων (A mazing workshop)	20
Πρόβλημα 4.1.3 Νησιά στη θάλασσα	21
Ημέρα 2η	22
Πρόβλημα 4.2.1. Το ρομπότ του Hamilton	22
Πρόβλημα 4.2.2. Σκαρφαλώνοντας το βουνό	23
Πρόβλημα 4.2.3. Η εργαλειοθήκη του Rubik	24
5η Διεθνής Ολυμπιάδα Πληροφορικής 1993	26
Πρώτος Γύρος	26
Πρόβλημα 1. Το περιδέραιο	26
Πρόβλημα 2. Οι εταιρείες	26
Πρόβλημα 3. Παραλληλόγραμμα στο χαρτί	27
Πρόβλημα 4. Διαγωνισμός αεροπορικής εταιρείας	27
6η Διεθνής Ολυμπιάδα Πληροφορικής 1994	29
Ημέρα 1η	29
Πρόβλημα 1.	29
Πρόβλημα 2. Κάτοψη Κάστρου	29
Πρόβλημα 3. Μαγικά τετράγωνα με 5ψήφιους πρώτους	30
Ημέρα 2η	31
Πρόβλημα 1.	31
Πρόβλημα 2. Σταθμός λεωφορείων	31
Πρόβλημα 3.	32
7η Διεθνής Ολυμπιάδα Πληροφορικής 1995	34
Ημέρα 1η	34
Πρόβλημα 1. Packing Rectangles	34
Πρόβλημα 2. Shopping Offers	34
Πρόβλημα 3. Printing	35
Ημέρα 2η	41
Πρόβλημα 1. Παιχνίδι με Γράμματα	41
Πρόβλημα 2. Αγώνας Δρόμου	41
Πρόβλημα 3. Καλώδια και Διακόπτες	42
8η Διεθνής Ολυμπιάδα Πληροφορικής 1996	44
Ημέρα 1η	44
Πρόβλημα 1. Το παιχνίδι	44
Πρόβλημα 2. Job Processing	44

Πρόβλημα 3. Το δίκτυο των σχολείων.....	45
Ημέρα 2η.....	45
Πρόβλημα 1. Ταξινομώντας μια τριπλέτα.....	45
Πρόβλημα 2. Longest Prefix	46
Πρόβλημα 3. Μαγικά τετράγωνα	47
9η Διεθνής Ολυμπιάδα Πληροφορικής 1997	48
Ημέρα 1η.....	48
Πρόβλημα 1. The Game of Hex.....	48
Πρόβλημα 2. Mars explorer.....	49
Πρόβλημα 3. The Toxic iShongololo	51
Ημέρα 2η.....	52
Πρόβλημα 1. Character Recognition	52
Πρόβλημα 2. Map labelling.....	54
Problem 3. Stacking containers	55
1η Βαλκανιάδα Πληροφορικής 1993.....	58
Ημέρα 1η.....	58
Πρόβλημα 1. Σχέδια στο χαρτί (40 πόντοι).....	58
Πρόβλημα 2. Master Mind (30 πόντοι)	58
Πρόβλημα 3. Προσθετική ακολουθία (30 πόντοι).....	58
Ημέρα 2η.....	58
Πρόβλημα 1.....	59
2η Βαλκανιάδα Πληροφορικής 1994.....	60
Ημέρα 1η.....	60
Πρόβλημα 1. (25 points)	60
Πρόβλημα 2. (40 πόντοι).....	60
Πρόβλημα 3. (35 πόντοι).....	61
Ημέρα 2η.....	61
Πρόβλημα 1.....	61
3η Βαλκανιάδα Πληροφορικής 1995.....	63
Day 1 - Problem 1 (Reservoirs)	63
Πρόβλημα 2. Μεγάλος Αριθμός	63
Ημέρα 2η.....	64
Πρόβλημα 1. Διανομές	64
Πρόβλημα 2. Πρωτάθλημα Baseball	64
Πρόβλημα 3. Περιστρεφόμενοι Κύλινδροι	65
4η Βαλκανιάδα Πληροφορικής 1996.....	66
Ημέρα 1η.....	66
Πρόβλημα 1. Το ενυδρείο	66
Πρόβλημα 2. Ο λαβύρινθος 1-2-3-4	66
Πρόβλημα 3. Κορώνα - Γράμματα	67
Ημέρα 2η.....	67
Πρόβλημα 1. Περιμένοντας για εισητήριο	68
Πρόβλημα 2. (Satellite configuration).....	68
Πρόβλημα 3. Παράθυρα σε ένα γραφικό περιβάλλον εργασίας.....	69
Central European Olympiads in Informatics	71
1st CEOI 1994.....	71
Ημέρα 2η.....	71
Πρόβλημα 1. Μαύρο - άσπρο.....	71
Πρόβλημα 2. Μονό – ζυγό δίκτυο.....	71
Πρόβλημα 3. Αριθμητικές Παραστάσεις	72
2nd CEOI 1995.....	73
Ημέρα 1η.....	73
Problem 1. Idle machine.....	73
Problem 2. Alarm chain	73
Problem 3. Fair sharing	74
Ημέρα 2η.....	74
Πρόβλημα 1. Ο ΚΗΠΟΣ.....	74
Πρόβλημα 2. ΤΟ ΠΑΡΤΥ	75
Πρόβλημα 3. ΤΑ ΠΟΤΗΡΙΑ ΜΕΤΡΗΣΗΣ	75
3rd CEOI 1996	76
Day 1.....	76
Problem 1. ENCODING GRID (40 pts).....	76
Problem 2. SHIPS (30 pts).....	76
Problem 3. HIGHWAY TOLLS (30 pts).....	77
Day 2.....	78
Problem 1. BIN PACKING (40 points)	78
Problem 2. CUTTING RECTANGLES (40 points).....	78

Problem 3. ELECTRICIAN (20 points)	78
4th CEOI	80
Problem 1. The Cave	80
Problem 2. Hexadecimal Numbers	81
Problem 3. Integer Intervals	81
Problem 4. River Crossing	82
Problem 5. Shooting Contest	83
Problem 6. Dominoes	83
American Computer Science League	85
Program 1: Time Card (Junior Division, 5 points)	85
Program 2: Botchagaloop (Junior Division, 5 points)	85
Program 3: Digit Count (Juionr and Intermediate Divisions, 10 points)	85
Program 4: You Are El (Junior, Intermediate, Senior Divisions, 10 pts)	86
Program 5: A Balancing Act, Revisited (Intermediate and Senior Divisions, 10 points)	86
Program 6: The Biggest, Revisited (Intermediate and Senior Divisions, 10 points)	87
Program 7: Squarepoint (Intermediate and Senior Divisions, 10 points)	87
Program 8: Alphaddition (Senior Division, 10 points)	88
All Ireland School Programming Competition 1994	89
Round 1	89
Problem 1. Horizontal Histogram	89
Problem 2. Vertical Histogram	89
Problem 3. Well Ordered Numbers	89
Problem 4. Roman Numerals	90
Problem 5. Hotel Keys	90
All Ireland Schools Programming Competition 1994	91
Final Round	91
Day 1	91
Problem 1. String Compression	91
Problem 2. Factorials	92
Problem 3. Transformations	92
Day 2	93
Problem 1. Raucous Rockers	94
Problem 2. Greedy Gift Givers	94
Problem 3. The Cat in the Hat	95
All Ireland Schools Programming Competition 1995	97
Round 1	97
Problem 1. Factorials	97
Problem 2. Is this a Triangle	97
Problem 3. Common Letters	97
4. Word Chain	97
Problem 5. Number Triangle	98
All Ireland Schools Programming Competition 1996	99
Round 1	99
1. Maximin	99
2. Egg Timer	99
Problem 3. Remainder	99
Problem 4. Encryption	99
Problem 5. Come to Nothing	99
Final Round	100
Day 1	100
Problem 1. The In between Sum	100
Problem 2. Word Construction	100
Problem 3. Snail Trail	100
Day 2	102
Problem 1. Rotating Words	102
Problem 2. Partial Problems	102
Problem 3. Letter Selection	102
Problem 4. Cube Chaos (Extra Problem)	103
HS Computing Problems Archive	105
Polynomial Solver -- Difficulty $**1/2$ on up	105
Print a Generic Month -- Difficulty $*1/2$	105
Print a Specific Month -- Difficulty $**1/2$	105
Shuttle Puzzle [Piele] -- Difficult $***$	105
Abundant Numbers [Sidney Kravitz] -- Difficulty $**1/2$ of 5	106
Feeding Game [Don Holstein Piele] -- $****$ out of 5	106
Product Star [Jaime & Juan Poniachik] -- $***$ of 5	107
Circular Sequences [Mihai Badoiu & others] -- Difficulty $****$ of 5	107

Superprime Beef [Piele, et al.] -- Difficulty: ***/5	108
Superprimes	108
Repusprimes [Piele]	108
Milknim [Jude Turian, traditional] -- Difficulty: ***/5	109
Calculator Language [South Pacific Contest, 1992] -- Difficulty: ***/5	109
Factors and Factorials [NZ Contest, 1993, Division I] -- Difficulty: **/5	110
Fractional HP Calculators [Kolstad, 1997] Difficulty: ***/5	110
Russ Cox's problem	111
USACO 96 Fall Championship	112
Πρόβλημα 1. The Errant Physicist [New Zealand Contest, 1989]	112
Πρόβλημα 2. Hamming Codes [Traditional, Kolstad]	113
Πρόβλημα 3. Palindromic Squares [Kolstad]	113
Πρόβλημα 4. Window Area [IV Balkan Olympiad]	114
USACO 96 Winter Championship	116
Πρόβλημα 1. Milk Containers [Piele, 1997]	116
Πρόβλημα 2. Super Roman Numerals [Kolstad, 1997]	116
Πρόβλημα 3. Babylonian Fractions [Traditional]	117
Πρόβλημα 4. Electrical Engineering Lab [Kolstad, 1997]	118
Πρόβλημα 5. Planetary Timekeeping [Kolstad, 1997 after Heinlein]	121
1997 USA Computing Olympiad	123
Πρόβλημα 1. Γραμματοσχημα	123
Πρόβλημα 2. Runaround Numbers	123
Πρόβλημα 3. Humble Numbers	124
Πρόβλημα 4. Digit Pals	124
USACO 97 Fall Championship	126
Problem 1: SNAIL TRAIL [All Ireland Contest]	126
Πρόβλημα 2: Ο τετράγωνος Αχυρώνας	127
Πρόβλημα 3: Πρόλογοι με Λατινικούς Αριθμούς	127
Πρόβλημα 4: Το Βοδήλατο [Don Gillies; Adapted by Galperin, Burch, Kolstad, 1995]	128
USACO 98 Winter Championship	129
Πρόβλημα 1: Τι ώρα είναι; [Richard Forster -- British Olympiad]	129
Πρόβλημα 2: DUAL PALINDROMES	131
[from Mario Cruz (Colombia), by Hugo Riekeboer (Argentina?)]	131
Πρόβλημα 3: Μοσχαρίσια McNUGGETS [Hubert Chen]	132
Πρόβλημα 4: Ο φράχτης [Brian Dean]	133
USACO 98 Spring Championship	134
Problem 1: DIFFERENT DICE [Hal Burch]	134
Problem 2: FEED RATIOS [Similar to ACM Finals, 1998, forwarded by Dan Adkins]	136
Problem 3: THE TAMWORTH TWO [Richard Forster, BOI]	137
Problem 4: SUBSET SUMS [Recreational Mathematics Newsletter, Kolstad]	138
Problem 5: STRINGSOBITS [Kim Schrijvers]	138

1η Διεθνής Ολυμπιάδα Πληροφορικής 1989

Η πρώτη ολυμπιάδα πληροφορικής έγινε στο Πράβιετς της Βουλγαρίας, στις 16-19 Μαΐου 1989. Εξι προβλήματα παρουσιάστηκαν στην επιτροπή του 1989.

Πρόβλημα 1. Μεταθέσεις Κουτιών

(Το πρόβλημα αυτό επιλέχθηκε τελικά για τον διαγωνισμό)

Δίνονται $2 \times N$ κουτιά, βαλμένα σε σειρά το ένα δίπλα στο άλλο ($N \leq 5$). Δύο γειτονικά κουτιά είναι άδεια, και τα άλλα περιέχουν $N-1$ σύμβολα "A" και $N-1$ σύμβολα "B". Για παράδειγμα, για $N = 5$:

A	B	B	A			A	B	A	B
---	---	---	---	--	--	---	---	---	---

Κανόνες ανταλλαγής :

Το περιεχόμενο από δύο γειτονικά μη-κενά κουτιά μπορεί να μετακινηθεί στα δύο άδεια κουτιά, κρατώντας τη σειρά τους.

Σκοπός:

Να καταλήξουμε σε μία διάταξη, όπου όλα τα "A" είναι αριστερά από όλα τα "B", ασχέτως με τη θέση των δύο άδειων κουτιών.

Πρόβλημα:

Γράψτε ένα πρόγραμμα το οποίο :

- Μοντελοποιεί την ανταλλαγή κουτιών, όπου ο αριθμός των κουτιών και η αρχική κατάσταση τους, εισάγονται από το πληκτρολόγιο. Κάθε ανταλλαγή εισάγεται από τον αριθμό (από 1 μέχρι $N-1$) του πρώτου από τα δύο γειτονικά κουτιά που θα έρθουν στη θέση των άδειων κουτιών. Το πρόγραμμα πρέπει να βρίσκει την κατάσταση των κουτιών μετά την ανταλλαγή και να τη δείχνει.
- Δοθείσης μιας αρχικής κατάστασης πρέπει να βρίσκει (αν υπάρχει) τουλάχιστον μία στρατηγική ανταλλαγών, η οποία να καταλήγει στον σκοπό του προβλήματος. Ενα πλάνο περιλαμβάνει την αρχική κατάσταση και τις ενδιάμεσες καταστάσεις για κάθε βήμα.
- Να βρίσκει το μικρότερο δυνατό πλάνο που καταλήγει στον σκοπό.

Πρόβλημα 2. Ανελκυστήρες σε κτίριο

Τα πατώματα σε ένα κτίριο είναι αριθμημένα με τη σειρά από ακέραιους αριθμούς 0, 1, 2, ..., N ($N \leq 15$). Υπάρχουν K ($1 \leq K \leq 4$) ανελκυστήρες στο κτίριο. Ο έλεγχος των ανελκυστήρων είναι κεντρικός και δέχεται δύο είδη κλήσεων, που εισάγονται πατώντας πλήκτρα. Τα εξωτερικά πλήκτρα (ένα για ζήτηση ανόδου - ένα καθόδου) υπάρχουν σε κάθε όροφο και είναι κοινά για όλους τους ανελκυστήρες. Τα εσωτερικά πλήκτρα για μετακίνηση σε συγκεκριμένο όροφο υπάρχουν σε κάθε ανελκυστήρα.

Γράψτε ένα πρόγραμμα το οποίο να μοντελοποιεί τον έλεγχο των ανελκυστήρων με τους ακόλουθες προϋποθέσεις :

- Υπάρχει ένας μόνο ανελκυστήρας στο κτίριο και μπορεί να δέχεται μία μόνο κλήση τη φορά. Κάθε άλλη κλήση γίνεται δεκτή μόνο αφού έχει ολοκληρωθεί η πρώτη.
- Υπάρχουν πολλοί ανελκυστήρες στο κτίριο ($K \geq 1$). Κάθε ένας δέχεται εσωτερική κλήση, μόνο αν δεν εκτελεί άλλη. Η κεντρική συσκευή ελέγχου μπορεί να δεχθεί πολλές κλήσεις την ίδια στιγμή. Οι εσωτερικές κλήσεις ολοκληρώνονται από τους ανελκυστήρες που έγιναν. Η κεντρική συσκευή ελέγχου, επιλέγει τον κατάλληλο ανελκυστήρα για να εκπληρώσει κάθε εξωτερική κλήση.
- Στην ίδια περίπτωση με τη 2, με τον περιορισμό, ότι οι περιττοί ανελκυστήρες σταματούν μόνο στους ορόφους με περιττό (μονό) αριθμό και οι άρτιοι ανελκυστήρες, στους ορόφους με άρτιο (ζυγό) αριθμό. Όλοι οι ανελκυστήρες σταματούν στον όροφο 0 (ισόγειο).
- Στην ίδια περίπτωση με την 3, και θεωρήστε ότι μπορούν να γίνονται πολλές εσωτερικές κλήσεις σε κάθε ανελκυστήρα, όχι μόνο μία. Όλες οι εσωτερικές κλήσεις, γίνονται δεκτές και καταχωρούνται, ασχέτως με το αν ο ανελκυστήρας είναι ελεύθερος ή όχι.

Πρόσθετες οδηγίες

Μπορείτε να δεχθείτε ότι οι ανελκυστήρες είναι συγχρονισμένοι, και σε ίσα χρονικά διαστήματα (ticks ρολογιού) κάθε ανελκυστήρας βρίσκεται σε συγκεκριμένο όροφο. Στη διάρκεια του επόμενου tick, ο ανελκυστήρας θα πάει έναν όροφο επάνω ή κάτω, ή θα παραμείνει στον ίδιο όροφο. Οι κλήσεις (είσοδος στο πρόγραμμα) θα μπορεί να γίνεται σε κάθε tick και είναι των ακόλουθων τύπων :

- Εξωτερικές - <όροφος, διεύθυνση κίνησης (πάνω/κάτω)>
- Εσωτερικές - <αριθμός ανελκυστήρα, αριθμός ορόφου>

- Πολλές ή καμία κλήσεις μπορούν να εισαχθούν σε κάθε tick.
- Σε κάθε tick το πρόγραμμα πρέπει να τυπώνει τη θέση του κάθε ανελκυστήρα.
- Οι ανελκυστήρες είναι αρκετά μεγάλοι και δεν γεμίζουν ποτέ.
- Το πρόγραμμα πρέπει να ελέγχει τους ανελκυστήρες έτσι ώστε η συμπεριφορά τους να δείχνει όσο το δυνατόν μεγαλύτερη "ευφυΐα".
- Θα πρέπει να υπάρχουν σαφείς ερμηνίες για τη στρατηγική ελέγχου των ανελκυστήρων.

Πρόβλημα 3. Το βιβλίο που θα διαβάσουν όλοι

Δίνεται μία ομάδα από N άτομα. Κάθε ένας είναι φίλος με όχι περισσότερους από $\lfloor N/2 \rfloor$ (ακέραιο μέρος της διαίρεσης) από τους υπόλοιπους και δεν έχει παραπάνω από K εχθρούς. Ένα από τα άτομα έχει ένα βιβλίο που όλοι θέλουν να το διαβάσουν και μετά να το συζητήσουν με κάποιους άλλους.

Γράψτε ένα πρόγραμμα το οποίο :

1. Βρίσκει έναν τρόπο να "περάσει" το βιβλίο από όλους, έτσι ώστε κάθε ένας να το πάρει μία μόνο φορά και να το δώσει σε κάποιον φίλο του, ώστε τελικά το βιβλίο να καταλήξει στον ιδιοκτήτη του.
2. Διαιρεί τα άτομα σε S υποομάδες για να συζητήσουν για το βιβλίο. Κάθε ένας δεν πρέπει να έχει περισσότερους από P εχθρούς στην ομάδα που ανήκει.

Εννοείται ότι $S \cdot P \geq K$.

Πρόβλημα 4. Κωδικοποίηση για Επικοινωνίες

Ας θεωρήσουμε μηνύματα τα οποία γράφονται χρησιμοποιώντας μόνο τα κεφαλαία γράμματα $A..Z$ και τα οκτώ σύμβολα $., + - : / ? !$

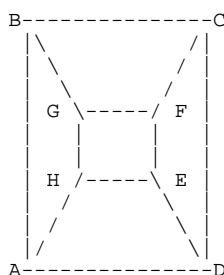
Τα μηνύματα αυτά στέλνονται μέσω ενός καναλιού επικοινωνίας σαν σειρές από bytes. Ο αριθμός από 1 σε κάθε byte πρέπει να είναι άρτιος (ζυγός).

1. Προτείνετε μία κωδικοποίηση για τους παραπάνω χαρακτήρες με δυαδικό συμβολισμό, έτσι ώστε να γίνεται σαφής αποκωδικοποίηση και να στέλνεται ο μικρότερος αριθμός από bits στο κανάλι.
2. Γράψτε ένα πρόγραμμα το οποίο:
 - a. Να του δίνεται ένα κείμενο και να τυπώνει την κωδικοποιημένη μορφή του, έτοιμη να αποσταλεί σε δεκαεξαδική μορφή.
 - b. Να του δίνεται ένα κωδικοποιημένο μήνυμα και να το αποκωδικοποιεί, δείχνοντας το πρωτότυπο κείμενο.

Πρόβλημα 5. Αριστερός και Δεξιός γείτονας

Ας θεωρήσουμε ένα επίπεδο γράφημα με n κορυφές.

Παράδειγμα:



Εστω οι κορυφές X, Y και Z είναι παρακείμενες της T . Θα λέμε ότι η Y είναι αριστερός γείτονας και η Z δεξιός γείτονας του T , συναρτήσει του X , αν η προσανατολισμένη γωνία XTZ είναι μικρότερη της γωνίας XTY (θετική είναι στη φορά την αντίθετη των δεικτών του ρολογιού).

Για παράδειγμα, η E είναι δεξιός και η G αριστερός γείτονας της H συναρτήσει της A γιατί η προσανατολισμένη γωνία AHE είναι μικρότερη από την AHG .

Γράψτε ένα πρόγραμμα το οποίο :

1. Εισάγει τις συντεταγμένες των κορυφών και των ακμών και ζωγραφίζει το σχήμα στην οθόνη του υπολογιστή, χρησιμοποιώντας κατάλληλη κλίμακα. (Οι ακμές πρέπει να είναι ευθείες γραμμές).
2. Να δέχεται ένα ζεύγος κορυφών X_0 και X_1 και μία ακολουθία από γράμματα L και R , και να βρίσκει μία διαδρομή $X_0X_1X_2...X_n$ στο γράφημα, τέτοια ώστε :
 - τα X_0 και X_1 να είναι οι δύο πρώτες κορυφές
 - X_{i+1} να είναι ο αριστερός ή δεξιός γείτονας του X_i συναρτήσει του X_{i-1} , αντιστοίχως του αν το $(i-2)$ στο γράμμα της ακολουθίας είναι L ή R .

Για παράδειγμα:

Η διαδρομή που θα παραχθεί από το παραπάνω γράφημα, χρησιμοποιώντας τις A και H σαν αρχικές κορυφές και την ακολουθία $LRLLR$ είναι $AHGFEDCB$.

3. Ζωγραφίζει το γράφημα που βρέθηκε στο υποπρόβλημα 2 στην οθόνη.
4. Χρησιμοποιώντας μια αρχική και μία τελική κορυφή, βρίσκει μία διαδρομή η οποία περνά από τον μικρότερο δυνατό αριθμό κορυφών, τη ζωγραφίζει και τυπώνει τις δύο αρχικές κορυφές και την ακολουθία από L και R, η οποία θα τη δημιουργούσε, όπως στο υποπρόβλημα 2.

Πρόβλημα 6. Εικοσάεδρο

Δίνεται ένα εικοσάεδρο. Είναι ένα κανονικό πολύεδρο. Οι πλευρές του είναι αριθμημένες από 1 μέχρι 20. Το εικοσάεδρο πρέπει να "περπατηθεί" έτσι ώστε να περάσουμε από κάθε πλευρά του ακριβώς μία φορά.

Το κόστος της διαδρομής C ορίζεται σαν :
$$C = \sum_{i=1}^{20} i * f_i$$
 όπου f_i είναι ο αριθμός της πλευράς που

βρισκόμαστε στο i βήμα.

Μπορούμε να περάσουμε από τη μία έδρα στην άλλη, μόνο αν οι έδρες αυτές είναι διπλανές.

A. Δύο έδρες είναι διπλανές όταν έχουν μία κοινή ακμή.

B. Δύο έδρες είναι διπλανές όταν έχουν μία κοινή ακμή ή ένα κοινό σημείο.

Βρείτε τις διαδρομές με το μικρότερο κόστος για τις δύο παραπάνω περιπτώσεις.

Σημείωση:

Αν λόγω χρόνου ή μεγέθους της πολυπλοκότητας του αλγορίθμου, δεν βρείτε την ακριβή λύση, μπορείτε τουλάχιστον να προτείνετε μία επαρκώς ικανοποιητική.

2η Διεθνής Ολυμπιάδα Πληροφορικής 1990

Η δεύτερη ολυμπιάδα πληροφορικής έγινε στο Μινσκ της Λευκορωσίας.

Γύρος 1ος.

Πρόβλημα 1. Puzzle με αριθμούς

Ενα πρόγραμμα το οποίο πραγματοποιεί την μεταμόρφωση του σχήματος A στο σχήμα B. Ένας αριθμός σε κάποιο κελί μπορεί να μεταφερθεί σε οποιοδήποτε από γειτονικά άδεια κελιά. Στην περίπτωση αυτή το κελί που καταλάμβανε ο αριθμός γίνεται κενό.

7	3	5	14
	4	9	13
1		2	10
11	8	12	6

A

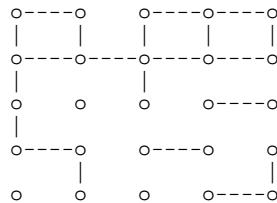
1	2	3	4
5	6	7	8
9	10	11	12
13	14		

B

Πρόβλημα 2. Παιχνίδι με Κουκίδες

Στο παιχνίδι με τις "κουκίδες", οι παίκτες ενώνουν τελείες οι οποίες είναι σχεδιασμένες οριζόντια και κάθετα σε ίσες αποστάσεις. Ο παίκτης ο οποίος ολοκληρώνει ένα τετράγωνο κερδίζει έναν πόντο και ξαναπαίζει. Στο τέλος του παιχνιδιού όταν όλες οι γραμμές έχουν σχεδιαστεί, νικητής είναι ο παίκτης με τους περισσότερους πόντους. Οι παίκτες μπορούν να σχεδιάσουν μόνο οριζόντιες και κάθετες γραμμές μεταξύ δύο κουκίδων.

Το παρακάτω παράδειγμα δείχνει ένα παιχνίδι μεταξύ Κόκκινων (Red) και Μπλέ (Blue) που είναι μισοτελειωμένο :



Το ίδιο σχήμα μπορεί να παρασταθεί και με R & B, δείχνοντας τις γραμμές που έχει σχεδιάσει ο κάθε παίκτης. Με O συμβολίζεται μία κενή γραμμή :

```
      R      O      B      R
B      R      B      B      B
B      B      B      R      R
R      O      R      O      O
O      O      O      B      O
B      O      O      O      O
B      O      R      O      O
O      R      O      O      R
O      O      O      B
```

Το παραπάνω μπορεί να "συμπίεστεί", σαν τον πίνακα :

```
ROBR
BRBBB
BBRR
ROROO
OOOB
BOOOO
BORO
OROOR
OOOB
```

1. Γράψτε ένα πρόγραμμα το οποίο "ψάχνει" έναν πίνακα σαν τον παραπάνω (ο οποίος πάντα αναπαριστά ένα παιχνίδι με 5 x 5 τελείες) και βρίσκει τον αριθμό των κουτιών με τρεις πλευρές (σε κάθε κατεύθυνση). Τα δεδομένα διαβάζονται από ένα αρχείο σαν μια σειρά από 9r γραμμές που αποτελούν r παιχνίδια και το οποίο τελειώνει με τη λέξη END σε μία ξεχωριστή γραμμή. Το πρόγραμμα πρέπει να τυπώνει μία γραμμή για κάθε πίνακα δεδομένων ως εξής:

```
MATRIX r contained x 3-sided box(es).
```

Που σημαίνει ότι ο πίνακας r περιέχει x κουτιά με τρεις γραμμές.

2. Συνεχίστε κάθε παιχνίδι για την περίπτωση του μπλε παίκτη. Συμπληρώστε όσα πιο πολλά τετράγωνα μπορείτε με μία κίνηση (θυμηθείτε ότι η ολοκλήρωση ενός τετραγώνου σημαίνει μία ακόμη κίνηση). Η τελική κίνηση είναι άσχετη, αλλά δεν θα πρέπει να καταλήγει σε ένα τετράγωνο με τρεις πλευρές εκτός κι αν είναι αναπόφευκτο. Το πρόγραμμα πρέπει να τυπώνει :

```
x boxes have been completed. Final move = L,C
```

όπου L και C είναι η γραμμή και η στήλη της τελευταίας κίνησης στον πίνακα και x είναι ο αριθμός των ΝΕΩΝ κουτιών που ολοκληρώθηκαν, όχι ο συνολικός αριθμός των ολοκληρωμένων κουτιών του πίνακα.

Πρόβλημα 3. Βιβλία για δύο αναγνώστες

Υπάρχουν N βιβλία και δύο αναγνώστες, A και B, που θέλουν να τα διαβάσουν. Δίνονται οι μη αρνητικοί αριθμοί A[I] και B[I] σαν οι ώρες που χρειάζονται αντίστοιχα οι αναγνώστες A και B για να διαβάσουν το I βιβλίο, $1 \leq I \leq N$. Και οι δύο αναγνώστες ξεκινούν την ώρα 0.

Κάθε στιγμή, κάθε αναγνώστης μπορεί να διαβάζει το πολύ ένα βιβλίο και δεν μπορούν και οι δύο να διαβάζουν το ίδιο βιβλίο. Δίνεται ο ακέραιος K, $2 \leq K \leq N$, και τα βιβλία είναι αριθμημένα κατά τέτοιο τρόπο ώστε δεν μπορεί κάποιος αναγνώστης να αρχίσει να διαβάζει το βιβλίο J, $2 \leq J \leq K$, έως ότου το βιβλίο J-1 να έχει διαβαστεί και από τους δύο.

H σειρά που διαβάζουν τα άλλα βιβλία, οι δύο αναγνώστες μας είναι αδιάφορη και μπορεί να είναι και αυθαίρετη.

Διακοπές ανάγνωσης. Είναι δυνατόν η διαδικασία ανάγνωσης ενός βιβλίου να διακοπεί σε κάθε χρονική ακέραια στιγμή και να συνεχιστεί αργότερα από το σημείο που σταμάτησε. Μεταξύ διακοπής και επανακίνησης ανάγνωσης ενός βιβλίου, ο αναγνώστης μπορεί να διαβάσει οποιοδήποτε άλλο βιβλίο που δεν έχει ολοκληρώσει και έχει το δικαίωμα να το διαβάσει.

Είναι απαραίτητο :

1. Να οργανώσετε την είσοδο των δεδομένων στη μορφή :

```
<ENTER N → >
<ENTER K → >
<ENTER: >
<A[1] → > <B[1] → >
<A[2] → > <B[2] → >
.....
<A[N] → > <B[N] → >
```

2. Να βρείτε το μεγαλύτερο δυνατό χρόνο T που δεν μπορούν να έχουν διαβαστεί τα βιβλία και από τους δύο αναγνώστες και να το τυπώσετε.
3. Να δημιουργήσετε ένα πρόγραμμα ανάγνωσης για κάθε αναγνώστη, το οποίο να συμβαδίζει με όλες τους παραπάνω περιορισμούς και κάτω από το οποίο η διαδικασία σταματά στον χρόνο T. Το πρόγραμμα θα τυπώνεται στη μορφή :

```
<SCHEDULE FOR READER A (or B) >
<Book> <Start> <Finish>
.....
.....
```

Στους παραπάνω πίνακες, πρέπει να φαίνονται όλα τα διαστήματα στα οποία ο αναγνώστης A (ή B) διαβάζει ένα βιβλίο και ο αριθμός του βιβλίου.

4. Τυπώστε τον αριθμό των διακοπόμενων αναγνώσεων για κάθε αναγνώστη. Προσπαθήστε να τον μειώσετε.

Πρόβλημα 4. Παιχνίδι για δύο παίκτες με συνεχόμενα κελιά

Δίνεται ένας ακέραιος K. Μία λωρίδα χαρτί είναι διαιρεμένη σε N κελιά ($K \leq N \leq 40$). Δύο παίκτες επιλέγουν και διαγράφουν K άδεια συνεχόμενα κελιά, ο ένας μετά τον άλλον. Νικητής είναι αυτός που κάνει την τελευταία κίνηση.

1	2											N

1. Διαβάστε το N και ορίστε αν ο παίκτης 1 έχει μία νικητήρια στρατηγική, που σημαίνει ότι μπορεί να κερδίσει ακόμα και με τις καλύτερες επόμενες κινήσεις του παίκτη 2). Τυπώστε το μήνυμα "Player 1 has winning strategy" ή "Player 1 doesn't have winning strategy".
2. Ορίστε για ένα δεδομένο N, αν ο παίκτης 1 έχει νικητήρια στρατηγική, αν η πρώτη του κίνηση εισάγεται στον υπολογιστή από το πληκτρολόγιο.
3. Φτιάξτε το παιχνίδι για δοσμένο N (παράγραφοι 1 και 2) και την πρώτη κίνηση του παίκτη 1. Προγραμματίστε κινήσεις για τον παίκτη 2. Οι κινήσεις του παίκτη 1, εισάγονται από το πληκτρολόγιο. Η κίνηση δίνεται από τη θέση του κελιού L ($1 \leq L \leq N-K+1$). Τα κελιά από το L μέχρι L+K-1 διαγράφονται κατά την κίνηση αυτή. Μετά από κάθε κίνηση, η κατάσταση του παιχνιδιού τυπώνεται ως εξής :

1 2* 3* N

όπου τα διαγεγραμμένα κελιά έχουν μαρκαριστεί με έναν αστερίσκο (*).

Πρέπει να τυπώσετε "Victory of Player 1(Player 2)" όταν τελειώσει το παιχνίδι. Όταν ζητάτε τα N και K, τυπώστε τα μηνύματα "N>" και "K>".
Όταν ζητάτε την κίνηση τυπώστε "Move of Player 1>".
Κάντε έλεγχο των εισαγόμενων δεδομένων.

Πρόβλημα 5. Γλώσσα PROLAN/M

Η εταιρεία NePhihhan hardware έχει φτιάξει έναν νέο μικρο-επεξεργαστή RISC, ικανό να ελέγχει έναν συγκεκριμένο τύπο δεδομένων - string χαρακτήρων - και να εκτελεί μία συγκεκριμένη λειτουργία στα strings, να κάνει αντικατάσταση χαρακτήρων (ψάξιμο για μια ακολουθία χαρακτήρων (substring) μέσα σε ένα string και αντικατάστασή των με μία άλλη ακολουθία χαρακτήρων). Χρησιμοποιούνται δύο περιοχές μνήμης, μία που περιέχει το πρόγραμμα (μία λίστα από περιγραφές πιθανών αντικαταστάσεων), και μία άλλη (που εμείς ονομάζουμε R και το μέγεθος της είναι πρακτικά χωρίς όρια) η οποία χρησιμοποιείται για να κρατά τα δεδομένα εισόδου, τα ενδιάμεσα αποτελέσματα και το τελικό αποτέλεσμα.

Τα προγράμματα για τον μικρο-επεξεργαστή γράφονται σε μία γλώσσα που ονομάζεται PROLAN/M. Πριν την περιγράψουμε φορμαλιστικά, θα δώσουμε ένα παράδειγμα ενός προγράμματος :

```
(aa,b) (ba,a) (bc,a) (c,start) (d,) (b,finish) (,)
```

Όταν εκτελεστεί στο string "abcabcd", το πρόγραμμα επιστρέφει σαν αποτέλεσμα το string "finish", ενώ τα περιεχόμενα της R πέρνουν τις ενδιάμεσες τιμές :

abcabcd, aaabcd, babcd, abcd, aad, bd, b, finish σταδιακά.

Τώρα, για την ακριβή περιγραφή της σύνταξης της PROLAN/M (χρησιμοποιούμε "::=" για να δηλώσουμε ότι "ορίζεται σαν" και ":" για να δηλώσουμε το "ή") :

```
<'PROLAN/M'-program> ::= <substitut.sequence>(,)  
<substitut.sequence> ::= <substitution>:  
                        ::= <substitut.sequence><substitution>  
<substitution> ::= (<left part>,<right part>)  
<left-hand part> ::= <string>  
<right-hand part> ::= <string>:<empty>  
<string> ::= <string symbol>:<string><string symbol>  
<empty string> ::=  
<string symbol> ::= <any ASCII character except `,' , `)' >
```

αφού το string εισόδου "φορτωθεί" στην R, το πρόγραμμα εκτελείται με τον ακόλουθο τρόπο: Ο επεξεργαστής ψάχνει για το πρώτο <substitution> στο <substitut.sequence> για το οποίο το <left-hand part> είναι ένα substring του string στην περιοχή R. Αν το ψάξιμο είναι επιτυχές, το <right-hand part> του ίδιου <substitution> αντικαθιστά το αντίστοιχο substring στην R (το πιο αριστερό αν δεν είναι μοναδικό). Η διαδικασία αυτή επαναλαμβάνεται από την αρχή του προγράμματος με τη νέα τιμή της R, μέχρι το <left-hand part> στο <'PROLAN/M'-program> βρεθεί σαν η τρέχουσα τιμή της R, η οποία τότε θεωρείται σαν τελικό αποτέλεσμα και η εκτέλεση του προγράμματος σταματά.

Ζητούμενο 1. Γράψτε και βγάλτε τα λάθη από ένα πρόγραμμα της PROLAN/M το οποίο μετατρέπει ένα string της μορφής <nat.nr1>+<nat.nr2>=? (<nat.nr1> και <nat.nr2> είναι ακολουθίες από ψηφία δεκαδικού συστήματος που περιγράφουν φυσικούς αριθμούς) σε ένα string της μορφής <nat.nr1>+<nat.nr2>=<nat.nr3> που περιέχει μία μαθηματικώς σωστή πρόταση (τα <nat.nr1> και <nat.nr2> μένουν τα ίδια). Για παράδειγμα, το string 1990+123=? πρέπει να μετατραπεί στο 1990+123=2113 στο τέλος της εκτέλεσης. Αποθηκεύστε το πρόγραμμά σας σε ένα αρχείο που θα λέγεται SUM.PRM

Ζητούμενο 2. Γράψτε ένα διορθωτή λαθών για την PROLAN/M (debugger). Θα πρέπει να μπορεί να κάνει τα ακόλουθα :

- να ζητά το όνομα του αρχείου κειμένου που περιέχει το πρόγραμμα σε PROLAN/M.
- να ζητά τα αρχικά περιεχόμενα της R
- να εκτελεί τους μετασχηματισμούς του string εισόδου ανάλογα με το πρόγραμμα στο αρχείο
- να τυπώνει το αποτέλεσμα στην οθόνη
- είναι επιθυμητό να μπορεί να επιτρέπει tracing mode, δηλαδή εκτέλεση προγράμματος βήμα - βήμα.

Η βαθμολογία για το ζήτημα 1 θα εξαρτηθεί από τον αριθμό των <substitutions> στο <substitution sequence>, όπως επίσης και στην ταχύτητα με την οποία τα test που θα δοθούν από την επιτροπή θα εκτελεστούν. Επομένως μπορεί να θέλετε να παραδώσετε δύο εκδόσεις του προγράμματος, κάθε μία από τις οποίες θα είναι καλύτερη στο να ικανοποιήσει και διαφορετικό κριτήριο αξιολόγησης. Το πρόγραμμα του ζητήματος 1, θα βαθμολογηθεί χρησιμοποιώντας ένα προγραμματιστικό σύστημα που σχεδιάστηκε από την επιτροπή γι' αυτόν τον σκοπό.

Η βαθμολογία για το ζήτημα 2, θα εξαρτηθεί από το αν “περάσει” τον έλεγχο της επιτροπής και από το αν έχετε ολοκληρώσει όλα υπο-ζητήματα.

Πρόβλημα 6. Υψωση σε δύναμη

Δίνονται ακέραιοι a και n ($n < 100$). Ας υποθέσουμε ότι μία φανταστική γλώσσα προγραμματισμού έχει μία πρόταση αντικατάστασης και έναν τελεστή πολλαπλασιασμού. Γράψτε ένα πρόγραμμα που δημιουργεί το κείμενο a^n αυτή τη γλώσσα για τον υπολογισμό του $b = a^n$, με το μικρότερο δυνατό αριθμό πολλαπλασιασμών. Ένα παράδειγμα του κειμένου αυτού για $n = 13$, όπου κάθε ζεύγος από άγκιστρα $\{ \}$ περιέχει σχόλια είναι το παρακάτω :

```
X1:=a;           {=a}
X2:=X1*X1;       {=a^2}
X3:=X2*X2;       {=a^4}
X4:=X3*X1;       {=a^5}
X5:=X3*X3;       {=a^8}
X6:=X5*X4;       {=a^13}
b:=X6;
```

Γύρος 2ος

Πρόβλημα 1. Φύλακες σε γκαλερί

Κάθε φύλακας σε μια γκαλερί τέχνης έχει υπηρεσία για κάποιο συνεχόμενο χρονικό διάστημα. Η βάρδια δίνονται σαν ένα ζεύγος $[T1_i, T2_i]$ - η ώρα αρχής και τέλους της βάρδιας του φύλακα i . Παίρνοντας ένα πρόγραμμα από βάρδιες πρέπει να κάνετε τα παρακάτω :

- Να ελέγξετε αν υπάρχουν τουλάχιστον δύο φύλακες της γκαλερί για κάθε χρονική στιγμή της περιόδου $[0, EndTime]$. Αν η υπόθεση a δεν πληρείται :
- Βρείτε όλες τις περιόδους που οι φύλακες δεν είναι αρκετοί (λιγότεροι από δύο φύλακες σε βάρδια) και
- Βρείτε τον ελάχιστο αριθμό από πρόσθετους φύλακες με βάρδιες προκαθορισμένης διάρκειας που απαιτούνται ώστε να έχουμε ένα σωστό πρόγραμμα βαρδιών, δηλαδή ένα πρόγραμμα που να πληρεί την υπόθεση a .

ΕΙΣΟΔΟΣ: (όλες οι ώρες δίνονται σαν ακέραια λεπτά).

EndTime	Η ώρα που η βάρδια τελειώνει, δηλαδή η γκαλερί πρέπει να φρουρείται μέσα στην περίοδο $[0, EndTime]$.
N	ο αριθμός των φυλάκων.
T1[i]. T2[i], $i=1, \dots, N$	η έναρξη και η λήξη της βάρδιας του φύλακα i .
Length	η προκαθορισμένη διάρκεια βάρδιας για τους πρόσθετους φύλακες.

ΕΞΟΔΟΣ:

- Η απάντηση για την υπόθεση (a) στη μορφή “Yes/No”.
- Αν η προηγούμενη απάντηση είναι “No”, η λίστα με τα ζευγάρια (k,l) - η αρχή και το τέλος των περιόδων με ελλειπή φύλαξη και ο αριθμός των φυλάκων εκείνη την περίοδο (0 ή 1).
- Ο αριθμός των πρόσθετων φυλάκων και η λίστα με την έναρξη και λήξη της βάρδιας κάθε πρόσθετου φύλακα.

Πρόβλημα 2. Ευθύγραμμα τμήματα και τέμνουσα ευθεία

Δίδονται N ($N > 0$) ευθύγραμμα τμήματα στο επίπεδο με τις συντεταγμένες των άκρων τους σαν δύο ζεύγη $(x1_i, y1_i)$, και $(x2_i, y2_i)$, $1 \leq i \leq N$. Τα άκρα των τμημάτων ανήκουν στα τμήματα.

Ζητούμενα:

- Οργανώστε την είσοδο των δεδομένων ως εξής:

```
<Enter N-the number of segments :>
<Enter co-ordinates of i'th segment:>
x1[1]→ y1[1]→
x2[1]→ y2[1]→
x1[2]→ y1[2]→
.....
```
- Βρείτε μία ευθεία γραμμή η οποία να έχει κοινά σημεία με όσα περισσότερα τμήματα γίνεται. Κοινά σημεία θεωρούνται και τα άκρα των τμημάτων. Τυπώστε με αύξουσα σειρά τους αριθμούς των τμημάτων που έχουν κοινά σημεία με την ευθεία.

Πρόβλημα 3. Η κίνηση των ρομπότ

N Κόμβοι αριθμημένοι με $1, 2, \dots, N$ ($N \leq 50$) είναι συνδεδεμένοι με ένα δίκτυο από δρόμους, ο καθένας από τους οποίους έχει μήκος 1. Οι δρόμοι “κινούνται” σε διαφορετικά ύψη και ενώνονται μόνο στους κόμβους. Στην αρχική χρονική στιγμή 0, υπάρχουν M ρομπότ σε κάποιους από τους κόμβους ($M = 2$ ή 3).

Τα ρομπότ κινούνται συνέχεια από τον ένα κόμβο στον άλλο και μπορούν να αλλάξουν την κατεύθυνση τους μόνο όταν βρίσκονται στους κόμβους. Δεν επιτρέπεται να σταματήσουν. Η ταχύτητα του i -στου ρομπότ ισούται με $speed[i]$ ($speed[i] = 1$ ή 2). Τα ρομπότ ελέγχονται με τέτοιον τρόπο ώστε να ελαχιστοποιηθεί ο χρόνος που πρέπει να βρεθούν όλα τα ρομπότ στην ίδια θέση.

Πρέπει να βρείτε τον ελάχιστο χρόνο T μετά τον οποίο η συνάντηση των ρομπότ μπορεί να επιτευχθεί, και να τον τυπώσετε, ή διαφορετικά να πείτε ότι είναι αδύνατο να βρεθούν όλα τα ρομπότ στην ίδια θέση με $t \geq 0$.

Η είσοδος των δεδομένων θα γίνει με τον ακόλουθο τρόπο:

```
<Input N:>
<Input number of roads K:>
<Road 1 connects points:>
...
<Road K connects points:>
```

(τα ζευγάρια δίνονται σαν $I J$)

```
<Input number of robots M:>
<Input speed of robot 1:>
...
<Input speed of robot M:>
<Input initial position of robot 1:>
...
<Input initial position of robot M:>
```

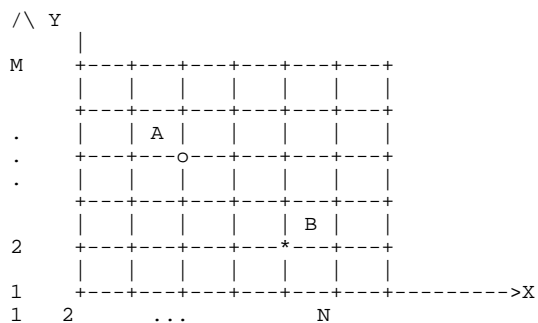
(Όλοι οι παραπάνω αριθμοί πρέπει να είναι μη-αρνητικοί αριθμοί).

Το αποτέλεσμα θα γραφεί:

```
<Time = ...>
```

Πρόβλημα 4. Ανηφόρες, κατηφόρες σε τετράγωνη πόλη

Όλοι οι δρόμοι σε μία ορθογώνιου σχήματος πόλη που βρίσκεται σε μία ανισοϋψή περιοχή, έχουν είτε την κατεύθυνση βορρά-νότου (N δρόμοι), είτε ανατολής- δύσης (M δρόμοι). Έτσι, η πόλη διαιρείται σε τετράγωνα κομμάτια με πλευρές ίσες με 1. Κάθε κομμάτι δρόμου μεταξύ δύο διασταυρώσεων, είναι ανηφόρα ή κατηφόρα ή επίπεδος δρόμος. Ο πίνακας $K[y,x]$ με διαστάσεις $M \times N$, περιέχει τα ύψη των διασταυρώσεων πάνω από το επίπεδο της θάλασσας.



Πρέπει να γράψετε ένα πρόγραμμα το οποίο:

1. Ζητά τις διαστάσεις του πίνακα - M και N .
2. Ζητά τα στοιχεία του πίνακα $H[i,j]$. $i=1..M$, $j=1..N$.
3. Ζητά τις συντεταγμένες δύο διασταυρώσεων - A και B .
4. Απαντά στην ερώτηση, αν είναι δυνατόν να μετακινηθούμε από το A στο B ή από το B στο A , κατηφορίζοντας συνεχώς. Αν η απάντηση είναι θετική, το πρόγραμμα πρέπει επίσης
5. Να βρίσκει τουλάχιστον μία τέτοια διαδρομή και να τυπώνει τις συντεταγμένες των διασταυρώσεων που περνάει
6. Να βρίσκει όλες τις διαδρομές αυτού του είδους.

3η Διεθνής Ολυμπιάδα Πληροφορικής 1991

Η τρίτη ολυμπιάδα πληροφορικής έγινε στην Αθήνα, στο διάστημα 19-25 Μαΐου 1991.

Πρόβλημα 1. Πασιέντσα με Τράπουλα

Μία τράπουλα αποτελείται από 52 χαρτιά, που έχουν 13 διαφορετικές τιμές 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K και τέσσερα διαφορετικά "χρώματα": Μπαστούνια, Σπαθιά, Καρώ και Κούπα (Τα μπαστούνια και τα σπαθιά είναι μαύρα, τα καρώ και οι κούπες είναι κόκκινα). Εχουμε μία τράπουλα και την ανακατεύουμε. "Τραβάμε" έξω από την τράπουλα 12 συνεχόμενα χαρτιά με τα οποία κάνουμε 3 γραμμές από 4 χαρτιά η κάθε μία, και τα τοποθετούμε πάνω σε ένα τραπέζι.

Σ' αυτή τη φάση, βγάζουμε πρώτα από την τράπουλα μία φιγούρα (J, Q ή K) και τη βάζουμε στο τέλος της τράπουλας, κατόπιν συνεχίζουμε να βγάζουμε τα 12 χαρτιά μέχρι να ολοκληρώσουμε το 3x4 σχήμα. Μετά, ελέγχουμε αν υπάρχουν ζευγάρια χαρτιών, σ' αυτά που ανοίξαμε τα οποία να μας δίνουν άθροισμα 10. (Αν υπάρχουν δύο 10άρια το ένα θεωρείται 0). Αν λοιπόν υπάρχουν τότε βγάζουμε για κάθε τέτοιο ζευγάρι, άλλα δύο χαρτιά από την τράπουλα και τα τοποθετούμε πάνω στα χαρτιά του ζευγαριού (ένα πάνω σε καθένα του ζευγαριού). Αν σ' αυτή τη φάση βγήκαν από την τράπουλα φιγούρες, τις αφήνουμε εκεί που είναι και τα χαρτιά αυτά δεν παίρνουν άλλο μέρος στη διαδικασία αυτή.

Η διαδικασία συνεχίζεται μέχρι όλα τα χαρτιά να σκεπαστούν από φιγούρες, ή να μην υπάρχουν "ανοιχτά" χαρτιά που κάποιο ζευγάρι να έχει άθροισμα 10.

Γράψτε ένα πρόγραμμα το οποίο προσομοιώνει την παραπάνω διαδικασία με τα ακόλουθα ζητούμενα:

1. Ο αριθμός N των επαναλήψεων της παραπάνω διαδικασίας, ζητείται από το πληκτρολόγιο. Αν το N είναι ίσο με 1, τότε κάθε ένα από τα παρακάτω ζητήματα 2, 3, και 4 εκτελείται.
2. a. Η τράπουλα δημιουργείται και για κάθε επανάληψη πρέπει να ανακατεύεται.
b. Η τράπουλα πρέπει να φαίνεται στην οθόνη.
3. a. Τα δώδεκα χαρτιά πρέπει να φαίνονται στην οθόνη στη μορφή που περιγράφτηκε.
b. Τα υπόλοιπα χαρτιά πρέπει να φαίνονται στην οθόνη.
4. a. Τα χαρτιά που είναι σκεπάζονται κατά την παραπάνω διαδικασία πρέπει να αντικαθίστανται με τα νέα χαρτιά που τα αντικαθιστούν και αυτό πρέπει να φαίνεται στην οθόνη.
b. Μετά από κάθε αντικατάσταση, τα χαρτιά που μένουν στην τράπουλα πρέπει να φαίνονται στην οθόνη.
5. Μετά από 5 επαναλήψεις, πρέπει να δείχνεται ένα ιστόγραμμα που θα φαίνονται ο αριθμός των χαρτιών που έμειναν στην τράπουλα μετά από το τέλος κάθε διαδικασίας.

Βαθμολογία :	1	10	βαθμοί
	2	(a) 15	βαθμοί
		(b) 5	βαθμοί
	3	(a) 10	βαθμοί
		(b) 5	βαθμοί
	4	(a) 10	βαθμοί
		(b) 5	βαθμοί
	5	15	βαθμοί
	Κώδικας	10	βαθμοί
	Επιτροπή	15	βαθμοί

Πρόβλημα 2. Η περιφραξη των δέντρων

Ενας αγρότης θέλει να διατηρήσει ένα σπάνιο είδος από αρχαία κωνοφόρα δέντρα. Για να το κάνει αυτό έχει κρατήσει σημειώσεις για τη θέση του κάθε δέντρου και σκοπεύει να περιφράξει τα δέντρα με σύρμα, σχεδιάζοντας ένα πολύγωνο έτσι ώστε τα δέντρα να βρίσκονται μέσα στην περιφραξη. Για να μειώσει το κόστος, πρέπει να ελαχιστοποιήσει το μήκος του σύρματος. Ο αγρότης επίσης θέλει να φτιάξει και ένα ορθογώνιο σπίτι, μία πλευρά του οποίου να είναι παράλληλη στο άξονα των x, και πρέπει να ξέρει τη θέση του σπιτιού :

1. Το σπίτι είναι έξω από το πολύγωνο.
2. Το σπίτι είναι μέσα στο πολύγωνο.
3. Η περιφραξη διαιρεί το σπίτι σε δύο κομμάτια, με εμβαδόν διαφορετικό του μηδενός (0).

Γράψτε ένα πρόγραμμα που να μπορεί να κάνει τα παρακάτω :

- a. Να βρει τα δέντρα τα οποία θα είναι οι κορυφές του πολυγώνου.
- b. Να υπολογίζει το μήκος του σύρματος που θα χρησιμοποιηθεί.
- c. Να βρίσκει σε ποιά από τις τρεις παραπάνω θέσεις βρίσκεται το σπίτι του αγρότη.

ΕΙΣΟΔΟΣ

N: Ο αριθμός των δέντρων.

$(X_i, Y_i) \ 1 \leq i \leq N, \ N \leq 20, \ X_i, \ Y_i > 0$. Οι συντεταγμένες των σημείων που αντιστοιχούν στο κάθε δέντρο.

$(a,b), (c,d), \ a,b,c,d > 0$. Η αρχή και το τέλος των σημείων της κυρίας διαγωνίου του σπιτιού.

ΕΞΟΔΟΣ

Μια ακολουθία από M σημεία ($1 \leq M \leq N$) με την ιδιότητα, αν "περπατήσουμε" μέσω των σημείων με τη σειρά που αυτά εμφανίζονται, να βρισκόμαστε πάνω στην περίμετρο της πολυγωνικής περιφράξης.

Το μήκος του σύρματος που θα χρησιμοποιηθεί.

Την ένδειξη "1", "2" ή "3" ανάλογα με τη θέση του σπιτιού σε σχέση με την περιφράξη.

Βαθμολόγηση: Είσοδος 10 πόντοι, A 40 πόντοι, B 10 πόντοι, C 30 πόντοι και 10 πόντοι στην κρίση της επιτροπής.

Πρόβλημα 3. Τετράγωνο (επιλέχθηκε για τον 1ο γύρο)

Θέλουμε να αριθμήσουμε κάθε θέση ενός 5x5 πίνακα, με τον ακόλουθο τρόπο: Αν ο αριθμός i ($1 \leq i \leq 25$) βρίσκεται στη θέση του πίνακα με συντεταγμένες (x,y), τότε ο αριθμός i+1 πρέπει να βρίσκεται σε μία από τις ακόλουθες θέσεις (z,w) :

$$(z, w) = (x+3, y)$$

$$(z, w) = (x, y+3)$$

$$(z, w) = (x+2, y+2)$$

Το πρόβλημα είναι :

Γράψτε ένα πρόγραμμα το οποίο να απαριθμεί τις θέσεις του 5x5 πίνακα, δεδομένης μίας αρχικής θέσης στην οποία βάζουμε τον αριθμό 1.

Υπολογίστε τον αριθμό όλων των δυνατών απαριθμήσεων για κάθε αρχική θέση στο πάνω δεξιό μέρος του πίνακα, συμπεριλαμβανομένης της κυρίας διαγωνίου.

Παράδειγμα:

Αν η θέση (2,2) επιλεγεί σαν αρχική, τότε ο αριθμός 2 μπορεί να μπει στις θέσεις με συντεταγμένες (2,5) ή (5,2) ή (4,4). Αυτές οι θέσεις είναι μαρκαρισμένες στο παρακάτω σχήμα με έναν αστερίσκο (*).

	1	2	3	4	5
1	:	:	:	:	:
2	:	:	1	:	:
3	:	:	:	:	:
4	:	:	:	:	:
5	:	:	:	:	:

Σημείωση: Θα εκτιμηθεί αρκετά αν το αποτέλεσμα του προγράμματος δείχνει σαν την εικόνα 1.

Βαθμολόγηση: A 50 πόντοι, B 25 πόντοι, Εξοδος 15 πόντοι και 10 πόντοι στην κρίση της επιτροπής.

Πρόβλημα 4. Οι ξένες γλώσσες

Δίνεται ένα ASCII αρχείο κειμένου το οποίο περιέχει κάποιο κείμενο γραμμένο σε μια φυσιολογική γλώσσα (π.χ. Αγγλικά, Γαλλικά, Γερμανικά κ.λ.π.). Η γλώσσα είναι άγνωστη, αλλά το χαρακτηριστικό της είναι ότι χρησιμοποιεί το λατινικό αλφάβητο.

Το πρόβλημα είναι:

Αναλύστε τα περιεχόμενα του αρχείου για να καταλάβετε σε ποιά γλώσσα είναι γραμμένο.

- Γράψτε ένα πρόγραμμα το οποίο θα διαβάζει τα περιεχόμενα του αρχείου και θα μετράει πόσοι χαρακτήρες περιέχονται σ' αυτό. Τυπώστε το αποτέλεσμα.
- Μετατρέψτε το πρόγραμμα (A) για να μετράει επίσης και τον αριθμό των επαναλήψεων κάθε γράμματος της αλφαβήτας, μετατρέποντας τους χαρακτήρες στίξης στον χαρακτήρα space (' ') και μετατρέποντας τα πεζά σε κεφαλαία. Ετσι μόνο οι χαρακτήρες που ανήκουν στο σύνολο [' ','A'..'Z'] θα πρέπει να μετρηθούν. Ταξινομήστε τους χαρακτήρες του συνόλου κατά αριθμό επαναλήψεων, με τον πιο συχνό χαρακτήρα πρώτο στη λίστα. Τυπώστε τη λίστα.
- Τροποποιήστε το πρόγραμμα (B) που μετράει τη συχνότητα των χαρακτήρων. Κανονικοποιήστε τις μετρήσεις. Δηλαδή, διαιρέστε τις επαναλήψεις του κάθε χαρακτήρα με τον συνολικό αριθμό των χαρακτήρων που διαβάστηκαν. Αυτό θα σας δώσει μια σχετική συχνότητα η οποία είναι ανεξάρτητη των συνολικών χαρακτήρων του κειμένου. Γράψτε τις σχετικές συχνότητες σε ένα αρχείο.
- Επεκτείνετε το πρόγραμμα (C) έτσι ώστε να διαβάσει ένα κείμενο και να το συγκρίνει με κάποια κείμενα από γνωστές γλώσσες. Η διαδικασία της σύγκρισης να γίνει με δική σας επινόηση. Σαν

αποτέλεσμα πρέπει να δίνει τη γλώσσα την οποία νομίζει το πρόγραμμα ότι είναι γραμμένο το άγνωστο κείμενο.

Σημείωση: Τα γνωστά κείμενα δίνονται με τα ονόματα ITA, FRA, κ.λ.π. για Ιταλικά, Γαλλικά κ.λ.π. αντίστοιχα. Το άγνωστο κείμενο έχει το όνομα TEXT.

Βαθμολόγηση: (A) 10 πόντοι, (B) 20 πόντοι, (C) 20 πόντοι, (D) 40 πόντοι και 10 πόντοι για την επιτροπή.

Δείγμα κειμένου:

Morire in conseguenza di anabolizzanti intrapresa per migliorare le proprie prestazioni atletiche 'e poi molto diverso dal morire al volante di un'auto di formula uno mentre si tenta di battere un record affrontando i rischi legati a quelle prove, o dal perire vittime di una scarica di sassi mentre si tenta una prima salita su una parete Nord di qualche colosso alpino? Che ci sia o no di mezzo la farmacologia, l'artificiale, il chimico, conta relativamente poco, sul piano essenziale, anche se sembra molto rilevante dal punto di vista dell'immaginario collettivo; il quale ancora considera spesso un eroe il pilota di auto da corsa o il grande alpinista, e ha invece un atteggiamento molto diverso nel caso di chi perisca vittima di un doping incauto. Naturalmente, qui entrano in gioco altri elementi, solo indirettamente morali. Fare uso di stimolanti chimici 'e in genere vietato nello sport, chi viola questa norma commette una grave slealtà: ma qui l'esecrazione morale non 'e motivata anzitutto dal fatto che viene messa a repentaglio l'integrità fisica dell'atleta, bensì dal mancato rispetto per regole del gioco. Le quali, per altro, vietano il doping anche e soprattutto per il danno fisico che esso alla lunga provoca; ma in base allo stesso principio, forse, dovrebbero vietare anche pratiche sportive "leali" e tuttavia non meno pericolose per chi le pratica..

Πρόβλημα 5. S-TERMS (επιλέχθηκε για τον 2ο γύρο)

Ένα s-term είναι μία ακολουθία από S και παρενθέσεις, η οποία ορίζεται αναδρομικά, ως εξής: το S είναι ένα s-term, και αν M και N είναι s-terms, τότε (MN) είναι επίσης s-term.

Παράδειγμα ενός s-term:

((((SS)(SS))S)(SS))

Οι δεξιές παρενθέσεις δεν δίνουν καμία νέα πληροφορία, οπότε μπορούν να παραληφθούν, δηλαδή (MN) αντί του (MN), έτσι ώστε το προηγούμενο s-term γράφεται:

((((SS(SSS(SS

1. Γράψτε μία διαδικασία 'gensterm' η οποία θα δημιουργεί s-terms: η διαδικασία θα πρέπει να δημιουργεί n (n=μήκος=αριθμός των S) αρχεία κειμένου που περιέχουν όλα s-terms μήκους 1,...,n αντίστοιχα. Τα S-terms πρέπει να χωρίζονται από ένα ";". Το τέλος κάθε s-term σε κάθε αρχείο θα πρέπει να σημειώνεται με ένα ".". Γράψτε ένα πρόγραμμα το οποίο δέχεται έναν ακέραιο n (<=10), χρησιμοποιεί την παραπάνω διαδικασία και δείχνει όλα τα δημιουργημένα s-terms στην οθόνη.

Θεωρίστε μία άλγεβρα με s-terms. Ο μόνος αλγεβρικός κανόνας (s-rule) ο οποίος μπορεί να χρησιμοποιηθεί είναι ο ακόλουθος: κάθε υπο-term ενός s-term το οποίο έχει τη μορφή ((SA)B)C (όπου A,B και C είναι s-terms) μπορεί να ξαναγραφτεί σαν: ((AC)(BC)) δηλαδή:

Context1(((SA)B)C)Context2 -> Context1((AC)(BC))Context2

Η εφαρμογή αυτού του κανόνα σε ένα s-term λέγεται "απλοποίηση" ενός s-term. Υπάρχουν διαφορετικοί τρόποι (στρατηγικές) για να επιλεγεί ένα υπο-term στο οποίο θα εφαρμόσουμε τον s-rule (τον κανόνα). Η συνεχής εφαρμογή του s-rule πάνω σε ένα s-term μέχρις ότου καμία άλλη εφαρμογή του κανόνα δεν θα είναι εφικτή, λέγεται "κανονικοποίηση" του s-term.

Παράδειγμα μιας αλυσίδας απλοποιήσεων είναι:

((((SS)(SS))S)(SS)) -> (((SS)((SS)S))(SS)) ->
((S(SS))((SS)S)(SS))) -> ((S(SS))((S(SS))(S(SS))))

2. Επιλέξτε μία κατάλληλη δομή δεδομένων για να αναπαραστήσετε τα s-terms η οποία μπορεί να δέχεται την εφαρμογή του s-rule. Γράψτε δύο διαδικασίες 'readterm' και 'printterm' οι οποίες μετατρέπουν s-terms στη (και από) δική σας αναπαράσταση από (και στη) δική σας αναπαράσταση τη μορφή που δημιουργήθηκε από την 'gensterm'. Το πρόγραμμα σας πρέπει να δείξει αυτές τις μετατροπές.
3. Γράψτε μία διαδικασία 'reduce' για να πραγματοποιεί ένα βήμα απλοποίησης, σύμφωνα με τον s-rule σε ένα οριζόμενο υπο-term ενός s-term που δίνεται στη δική σας αναπαράσταση. Το πρόγραμμα πρέπει να μπορεί να το δείξει αυτό.
4. Γράψτε μία διαδικασία 'normalize': δεδομένου ενός s-term, θα πρέπει να επιλέγει ένα υπο-term και να του εφαρμόζει τον s-rule, μέχρι να μην μπορεί να το κάνει πια ή μέχρι ο αριθμός των απλοποιήσεων ξεπεράσει κάποιο maximum, π.χ. 30. Το πρόγραμμά σας πρέπει να μπορεί να δείχνει την όλη λειτουργία.
5. Τέλος, ενσωματώστε όλα τα παραπάνω σε ένα πρόγραμμα, το οποίο:
 - a) ζητάει το μήκος n από τον χρήστη.
 - b) χρησιμοποιεί s-terms μήκους n, που δημιουργήθηκαν από την 'gensterm'
 - c) μετατρέπει s-terms στη δική σας αναπαράσταση
 - d) κανονικοποιεί (αν είναι δυνατόν)
 - e) τυπώνει τα τελικά (κανονικοποιημένα) s-terms
 - f) τυπώνει τον αριθμό των βημάτων απλοποίησης που έγιναν σε κάθε s-term, ή ένα μήνυμα 'not normalized' στην περίπτωση που δεν μπορεί να γίνει κανονικοποίηση σε 30 βήματα, και
 - g) τυπώνει τον αριθμό των μη-κανονικοποιημένων και τον συνολικό αριθμό των s-terms για το δεδομένο μήκος n.

Βαθμολόγηση: (1) 20 πόντοι, (2) 25 πόντοι, (3) 15 πόντοι, (4) 20 πόντοι, (5) 10 πόντοι και 10 πόντοι για την επιτροπή.

Πρόβλημα 6. Η μέγιστη συμμορία

A Police captain knows well all the outlaw persons of his city, as well as, every possible collaboration among them. He would like to find the maximum gang of the city.

In this case, a gang is a subset of outlaw persons so that any person in it collaborates with all others in the subset. Maximum gang means that there does not exist another gang with greater cardinality.

Create a program capable of carrying out the following tasks:

(A) Accept the police captain's data with a total number of outlaw persons less than 41. Use as input data file an ASCII one with the following structure:

```
a(1,1)
a(2,1)a(2,2)
a(3,1)a(3,2)a(3,3)
.....
a(n,1)a(n,2)a(n,3).....a(n,n)
```

where $a(i,j)=1$, if person i is collaborating with person j or $i=j$, and $a(i,j)=0$, otherwise. For instance (in the case of 6 persons):

```
1
01
101
1011
01101
101111
```

In this example an output can be the following one:

A. Maximum Gang is :

```
1      3 4 6
cardinality = 4
```

- B. Extend the input part of the program, so that to generate data in a random way under a given propability $0 < d < 1$ of collaboration of outlaw persons.
- C. Using random data or input file, find the maximum gang of the city. Use the same output format as in the example above (see task A).

EVALUATION: (A) 5 points, (B) 15 points, (C) 50 points, Total Machine Time 20 points, Jury 10 points.

Πρόβλημα 7. Τα ραντεβού του Ιατρικού Επισκέπτη

A medical visitor would like to have a program which would schedule all his medical appointments and help him meet as many doctors as possible in one day in order to advertise the medical products of the company he represents.

You are asked to write the program which would do this task for him. The input file consists of lines, each of which contains the name of a doctor, the beginning and the end of the interval when the doctor is ready to meet the medical visitor, as well as the name of his medical institute.

The rules for scheduling the appointments are as follows:

1. An appointment has at least 70 minutes duration; furthermore the medical visitor needs at least 30 minutes between any two appointments to travel to the next medical institute.
2. All the times in the input file belong to the same day, and the medical visitor does not want to meet any doctor twice on the same day.
3. Also, two consecutive appointments cannot be held at the same institute.
4. Among several doctors the medical visitor always prefers the one whom he can meet earlier.
5. If there is more than one doctor whom the medical visitor would be able to meet at the same time (either because by the time he finishes from the previous appointment there are doctors already waiting or because their proposed appointment would begin later but at the same time during the day) he prefers the one who has less remaining time for the given day (but of course this time at the beginning of the appointment must be enough for the required 70 minutes).
6. To visit as many doctors as possible, the medical visitor always (even during an appointment) keeps in mind the next appointment's starting time, therefore if it is necessary he can terminate the on-going appointment after the minimal 70 minutes expire, take his 30 minutes for traveling and begin the next appointment according to rule 5.
7. If he is not in a hurry for a new appointment (according to rule 6), the medical visitor stays with a doctor as long as he can.

Input file:

consists of lines each of which contains a name (with the English alphabet), a blank space , a time (in hh.mm form) indicating the possible beginning of an appointment, a dash ('-'), a time again indicating the last possible moment of the appointment, a blank space, and a name (with the English alphabet) of the medical institute.

The input file does not contain empty lines, nor is the end of it marked with special characters.

A possible input file is as follows:

```
Bob 16.00-17.25 Cross
John 09.30-11.50 Health
Charles 11.00-20.00 Chest
Don 08.00-13.20 Cross
Norman 22.00-23.05 Brain
Jerry 10.00-17.00 Health
Charles 09.20-10.40 Orthopedic
Evelyn 19.15-20.40 Orthopedic
Peter 09.35-11.55 Brain
Don 18.00-20.00 Eye
```

Output file:

should contain a table in which all the possible appointments appear in their proposed chronological order. The appointments which are ruled out by any of the rules above should not appear in the table.

The table should consist of lines containing the realizable beginning and end time of the possible appointments, their place, and the proposing doctor's name. The exact spacing of the output table is not important, but should look similar to the one below. The output for the previous input file is as follows:

08.00-09.10	Cross	Don
09.40-10.50	Health	John
11.20-12.30	Chest	Charles
13.00-15.30	Health	Jerry
16.00-17.25	Cross	Bob
19.15-20.40	Orthopedic	Evelyn

8. Solve the above problem for two medical visitors A and B preserving the same scheduling rules and the additional restriction that they are not allowed to visit the same doctor. Each time an appointment is available, it is taken by that medical visitor who has been free for the longest period. If both have been free for the same time period, Mr. A takes the appointment. The output consists of two appointment lists.

Evaluation: One medical visitor 45 points, Two medical visitors 45 points, Jury 10 points (elegance, clarity, progr. style)

[illegible]


```

[[[[[[[[[[[[[[[[[[
LENGTH = 13

```

Example-2: The same maze's file output by Tool-4 should look like:

```

10      8
[[[[[[[[[[[[  [[[[[[
[[[[[[[[  [[  [[
[[[[  [[  [[  [[
[[  [[  [[  [[  [[
[[  [[  [[  [[  [[
[[[[  [[  [[  [[[[
[[  [[T  [[
[[[[[[[[[[[[[[[[[[

```

CREDITS

```

Main menu with all tools available ..... 5 points
Tools available in any order and repetition ..... 10 points
Tool-1 enables setting N and M ..... 5 points
Tool-2 enables setting DELAY TIME ..... 5 points
Tool-3 computes structurally correct mazes ..... 30 points
Tool-3 displays the maze while it is growing ..... 10 points
Tool-4 writes maze to a file exactly as in example-2 ..... 5 points
Tool-5 reads unknown maze and highlights longest path ..... 20 points
Technical constraints completely obeyed ..... 10 points
-----
maximal 100 points

```

Problem Chosen for the first session (5 hours)

Πρόβλημα 4.1.3 Νησιά στη θάλασσα

The SEA is represented by an N times N grid. Each ISLAND is a "*" on that grid. The task is to reconstruct a MAP of islands only from some CODED INFORMATION about the horizontal and vertical distribution of the islands. To illustrate this code, consider the following map:

```

*   * *      1 2
*   * * *    3 1
*   *   *    1 1 1
*   * * * *   5
* *   *   *   2 1 1
      *       1

1 1 4 2 2 1
1 2   3   2
1

```

The numbers on the right of each row represent the order and size of the groups of islands in that rows. For example, "1 2" in the first row means that this row contains a group of one island followed by a group of two islands; with sea of arbitrary length to the left and right of each island group. Similarly, the sequence "1 1 1" below the first column means that this column contains three groups with one island each, etc.

Το πρόβλημα

Implement a program which repeats the following steps until a given input file containing several information blocks has been read completely:

1. Read the next information block from an ASCII input file (for the data structure of that file see also the examples below) and display it on the screen. Each information block consists of the size of the square grid, followed by the row constraints and the column constraints. Each constraint for a single row or column appears on a single line as a sequence of numbers separated by spaces and terminated by 0.
2. Reconstruct the map (or all of the maps, if more then one solution is possible, see Example-4) and display it/them on the screen.
3. Write the map(s) to the end of an ASCII output file. Each blank must be represented by a pair of spaces. Each island should be represented by a '*' followed by a space. Different maps satisfying the same constraints should be separated by a blank line. If there is no map satisfying the constraints, indicate it by a line saying "no map". The solutions to the different information blocks must be separated by a line saying "next problem".

Τεχνικοί Περιορισμοί

- Constraint-1: N must be not less than 1 and not larger than 8.

Παραδείγματα

```

6      Example-1 (the problem above): 6 is the size of the grid.
1 2 0      <-- The start of the first line constraint

```

```

3 1 0
1 1 1 0
5 0
2 1 1 0
1 0
1 1 1 0      <-- The start of the first column constraint
1 2 0
4 0
2 3 0
2 0
1 2 0

4      Example-2. Solution: columns: 1 2 3 4
0      row 1:
1 0      row 2:      *
2 0      row 3:      * *
0      row 4:
0
1 0
2 0
0

2      Example-3. Note that there is no map
0      satisfying the constraints.
0
2 0
2 0

2      Example-4. Note that there are two different maps
1 0      satisfying the constraints.
1 0
1 0
1 0

```

CREDITS

```

Read an information block from
the input file and display it ..... 5 points
Process all information blocks one by one
until the input file is read completely ..... 10 points
Reconstruct one map for each information
block (if it has a solution) and display it ..... 35 points
Write the solution map to the output file ..... 5 points
Reconstruct all possible maps (if there
are several solutions) and display them ..... 20 points
Write all solution maps correctly
separated to the output file ..... 10 points
Identify information blocks having no solution ..... 5 points
Technical constraints completely obeyed ..... 10 points
-----
maximal 100 points

```

Ημέρα 2η

Πρόβλημα 4.2.1. Το ρομπότ του Hamilton

Στο επίπεδο δίνονται N θέσεις P_1, P_2, \dots, P_N με ακέραιες συντεταγμένες $(X_1, Y_1), (X_2, Y_2), \dots, (X_N, Y_N)$.

Ενα ρομπότ πρέπει να μετακινηθεί σε όλες αυτές τις θέσεις ξεκινώντας από την P_1 . Πρέπει να περάσει από κάθε θέση μόνο μία φορά με εξαίρεση την P_1 η οποία πρέπει να είναι και η τελευταία θέση της "βόλτας" του.

Υπάρχουν όμως περιορισμοί στις κινήσεις του ρομπότ. Μπορεί μόνο να κινηθεί σε ευθείες γραμμές. Από τη θέση P_1 μπορεί να ξεκινήσει προς οποιαδήποτε κατεύθυνση. Όταν φτάσει σε κάποια θέση P_i , πριν μετακινηθεί σε κάποια άλλη, πρέπει να στρίψει 90 μοίρες είτε δεξιά, είτε αριστερά.

Το πρόγραμμα του ρομπότ απαρτίζεται από πέντε είδη προτάσεων:

1. "ORIENTATION $X_k Y_k$ ": χρησιμεύει μόνο σαν πρώτη πρόταση. Το ρομπότ στρίβει προς την κατεύθυνση της θέσης P_k (k μεταξύ 2 και N).
2. "MOVE-TO $X_j Y_j$ ": αν το ρομπότ μπορεί να φτάσει στο P_j χωρίς να αλλάξει τον προσανατολισμό του, τότε κινείται στη θέση P_j (j μεταξύ 1 και N). Σε άλλη περίπτωση η πρόταση δεν εκτελείται.
3. "TURN-LEFT": το ρομπότ στρίβει 90 μοίρες αριστερά.
4. "TURN-RIGHT": το ρομπότ στρίβει 90 μοίρες δεξιά.
5. "STOP": απενεργοποιεί το ρομπότ. Αυτή είναι η αναγκαία τελευταία πρόταση του προγράμματος.

Το πρόβλημα

Γράψτε ένα πρόγραμμα που να κάνει τα ακόλουθα:

1. Διαβάστε την τιμή του N και τις συντεταγμένες των N δοσμένων θέσεων από ένα ASCII αρχείο (δείτε το παράδειγμα) και τυπώστε τα δεδομένα στην οθόνη.
2. Φτιάξτε ένα πρόγραμμα για το ρομπότ, ώστε να κάνει μία σωστή "βόλτα" περνώντας από όλες τις θέσεις, αν υπάρχει τέτοια "βόλτα".
3. Αν δεν μπορεί να γίνει μία τέτοια "βόλτα", τότε το πρόγραμμα του ρομπότ πρέπει να αποτελείται μόνο από την πρόταση "STOP".
4. Τυπώστε στην οθόνη, το αν μπορεί να γίνει μία "βόλτα" ή όχι, και αν μπορεί τότε τυπώστε το μήκος της (στρογγύλευση σε 2 δεκαδικά ψηφία). Το μήκος της "βόλτας" είναι το άθροισμα των μηκών των κομματιών της τεθλασμένης γραμμής.
5. Γράψτε το πρόγραμμα του ρομπότ σε ένα ASCII αρχείο όπως φαίνεται στο παράδειγμα.

Τεχνικοί Περιορισμοί

- Program must reject inputs where N is less than 4 or greater than 16, without trying to find a tour!

Παράδειγματα

Input: An input file contains in the first line the value for N and in the following N lines the X and Y coordinates of the selected positions, for example:

```
4
2 -2
0 2
```

```
1 -1
3 1
```

Output: For these 4 positions one shortest robot program with length = 12.65 is:

```
ORIENTATION 3 1
MOVE-TO 3 1
TURN-LEFT
MOVE-TO 0 2
TURN-LEFT
MOVE-TO -1 -1
TURN-LEFT
MOVE-TO 2 -2
STOP
```

CREDITS

Read input data correctly from every file and display it.....	5 points
Algorithm for computing a valid tour ok	30 points
Generated robot program syntactically correct, if tour does not exist ..	10 points
Generated robot program syntactically correct, if tour does exist	15 points
Screen display gives all required information	5 points
Displayed length of computed tour correct	10 points
Robot program correctly written to a file	10 points
Technical constraints obeyed	15 points

maximal 100 points

Problem Chosen for the second session (5 hours)

Πρόβλημα 4.2.2. Σκαρφαλώνοντας το βουνό.

A mountain climbers club has P members, numbered from 1 to P. Every member climbs at the same speed and there is no difference in speed between climbing up and down. Climber number i consumes C(i) units of SUPPLIES per day and can carry at most S(i) such units. All C(i) and S(i) are integer numbers.

Assume that a climber with a sufficient amount of supplies would need N days to reach the top of the mountain. The mountain may be too high, so that a single climber cannot carry all the necessary supplies. Therefore a GROUP of climbers starts at the same place and at the same time. A climber who descends prematurely before reaching the top gives his unneeded supplies to other climbers. The climbers do not rest during the expedition.

The PROBLEM is to plan a schedule for the climbing club. At least one climber must reach the top of the mountain and all climbers of the selected group return to the starting point.

PROBLEM STATEMENT

=====

Implement a program which does the following:

1. Read from the keyboard the integer number N of days needed to arrive at the top, the number P of climbers in the club, and (for all i from 1 to P) the numbers S(i) and C(i). You may assume that the inputs are integers. Reject inputs that make no sense.
2. Try to find a schedule for climbing the mountain. Determine a possible group a(1), ..., a(k) of climbers who should participate in the party and (for all j from 1 to k) the number M(j) of supplies which

climber $a(j)$ carries at the start. Note that there may not exist a schedule for all combinations of N and the $S(i)$ and $C(i)$.

3. Output the following information on the screen:
 - a) the number k of climbers actually participating in the party,
 - b) the total amount of supplies needed,
 - c) the climber numbers $a(1), \dots, a(k)$,
 - d) for all $a(j)$, j between 1 and k , the initial amount $M(j)$ of supplies to carry for climber $a(j)$,
 - e) the day $D(j)$ when climber $a(j)$ starts descending.
4. A schedule is OPTIMAL if
 - a) the number of participating climbers is minimal and
 - b) among all groups satisfying condition a) the total of consumed supplies is minimal.Try to find a nearly optimal schedule.

TECHNICAL CONSTRAINTS

Constraint-1: Programs must reject inputs where N is less than 1 or greater than 100. P must be not less than 1 and not greater than 20.

Παραδείγματα

Τα παρακάτω θα μπορούσαν να είναι ένα διάλογος με το πρόγραμμά σας:

```
Days to arrive to top: 4
Number of club members: 5
Maximal supply for climber 1 : 7
Daily consumption for climber 1 : 1
Maximal supply for climber 2 : 8
Daily consumption for climber 2 : 2
Maximal supply for climber 3 : 12
Daily consumption for climber 3 : 2
Maximal supply for climber 4 : 15
Daily consumption for climber 4 : 3
Maximal supply for climber 5 : 7
Daily consumption for climber 5 : 1
```

```
2 climbers needed, total amount of supplies is 10.
Climber(s) 1, 5 will go.
Climber 1 carries 7 and descends after 4 day(s)
Climber 5 carries 3 and descends after 1 day(s)
```

```
Plan another party (Y/N) Y
Days to arrive to top: 2
Number of club members: 1
Maximal supply for climber 1 : 3
Daily consumption for climber 1 : 1
Climbing party impossible.
Plan another party (Y/N) N
```

Good bye

CREDITS

```
User dialogue as illustrated above..... 10 points
Find a solution for the special case where all  $C(i)=1$  and
all  $S(i)$  are equal ..... 20 points
Find a solution for general case ..... 20 points
Find a nearly optimal solution for general case ..... 30 points
Detect unsolvable situations ..... 10 points
Technical constraints obeyed ..... 10 points
-----
maximal 100 points
```

Πρόβλημα 4.2.3. Η εργαλειοθήκη του Rubik

This problem is based on the puzzle game "Rubik's cube". If you already know Rubik's cube you may skip this paragraph and the next one. Rubik's cube is a cube that consists of $3 \times 3 \times 3$ smaller cubes. Initially each of the six faces of Rubik's cube is coloured uniformly in a different colour; we call this the initial cube. Every face of Rubik's cube consists of 3×3 faces of a layer of nine smaller cubes.

Imagine you are looking at any of the six faces of Rubik's cube. The layer of 3×3 smaller cubes you see can be rotated by a multiple of 90 degrees, where the axis of rotation is orthogonal to the face and goes through its centre. The result is another $3 \times 3 \times 3$ cube where the colour pattern of the face you are looking at has been rotated and the colour patterns of the four neighbouring faces have changed.

In our problem the faces of the cube are given names instead of colours: U=Up, R=Right, F=Front, B=Back, L=Left and D=Down. Any move sequence to turn the cube may be described as a string of the letters {U, R, F, B, L, D} where each letter stands for a basic rotation: the 90 degrees clockwise rotation of the corresponding face.

PROBLEM STATEMENT with EXAMPLE(S)

Write a program that allows the user to repeatedly solve any of the given three subproblems in any order. You may assume that the length of each input string is at most 35.

1. This subproblem is the translation of a given move sequence into a move sequence where no primitive rotation is applied more than 3 times in sequence. Your algorithm should reject non-legal input sequences. Some examples are provided for clearness:

Input		Output
L	-->	L
LL	-->	LL
LLL	-->	LLL
LLLL	-->	"the empty sequence"
LLLLL	-->	L
LLRRRFFFFRLB	-->	LLLB
HELLO	-->	"error"

2. The second subproblem is to find out whether two given move sequences yield the same result when applied to the initial cube. The examples may illustrate this:

Input, 1 st sequence	Input, 2 nd sequence	Output
RL	LR	yes
RU	UR	no
RRFFRRFFRRFFRRFF	FFRRFFRR	yes
RRFFRRFFRRFFRRFF	RRFFRRFF	no

3. The third subproblem is to determine how many times a given move sequence has to be applied to the initial cube until the cube is in its initial state again. The smallest such number greater zero is sought.

We provide some examples:

Input	Output
L	4
DD	2
BLUB	36
RUF	80
BLUFF	180

CREDITS

Main menu and user dialogue o.k.	15 points
Subproblem 1: Transformation o.k.	20 points
Rejects wrong inputs	10 points
Subproblem 2: Correctness	25 points
Subproblem 3: Correctness	25 points
Technical constraints obeyed	5 points

maximal 100 points

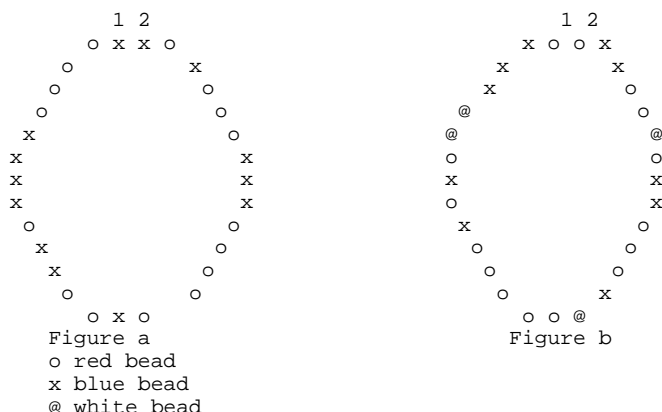
5η Διεθνής Ολυμπιάδα Πληροφορικής 1993

Η πέμπτη ολυμπιάδα πληροφορικής έγινε...

Πρώτος Γύρος

Πρόβλημα 1. Το περιδέραιο

You have a necklace of n beads ($n < 100$) some of which are red, others blue and others white, arranged at random. Let's see two examples for $n = 29$:



(The beads considered first and second in the text that follows have been marked in the picture). The configuration in Fig. a) may be represented as a string of b's and r's, where b represents a blue bead and r a red one, as follows: brbrrrbbrbrrrrrbrrbrrbbbrbrrrb

Suppose you are to break the necklace, lay it out straight, and then collect beads of the same colour from one end until you reach a bead of a different colour, and do the same for the other end (which may not be of the same colour as the beads collected before this).

Determine the point where the necklace should be broken so that the most number of beads can be collected.

For example, for the necklace in Fig. a), 8 beads can be collected, with the breaking point either between bead 9 and bead 10, or between bead 24 and bead 25.

In some necklaces, white beads had been included as shown in Figure b) above. When collecting beads, a white bead that is encountered may be treated as either red or blue, and painted with the desired colour. The string that represents this configuration will include the symbols: r, b and w.

Write a program to do the following:

1. Read a configuration from an ASCII input file, NECKLACE.DAT, with each configuration on one line. Write this data into an ASCII output file, NECKLACE.SOL. An example of an input file would be:

```
brbrrrbbrbrrrrrbrrbrrbbbrbrrrb  
bbwbrrrwbrbrrrrrb
```

2. For each configuration, determine the maximum number, M , of beads collectable, along with a breaking point.
3. Write to the outfile, NECKLACE.SOL, the number M and the breaking point. The solutions for different configurations should be separated with a blank record. Example of a possible solution:

```
brbrrrbbrbrrrrrbrrbrrbbbrbrrrb  
8 between 9 and 10
```

```
bbwbrrrwbrbrrrrrb  
10 between 16 and 17
```

Πρόβλημα 2. Οι εταιρείες

Some companies are partial owners of other companies because they have acquired part of their total shares. For example, Ford owns 12% of Mazda. It is said that a company A controls company B if, at least, one of the following conditions is satisfied:

A = B

A owns more than 50% of B

A controls k ($k > 1$) companies, $C(1), \dots, C(k)$, so that: $C(i)$ owns $x(i)\%$ of B for $1 < i < k$ and $x(1) + \dots + x(k) > 50$.

The problem to solve is:

Given a list of triples (i,j,p) which means that the company i owns p% of company j, calculate all the pairs (h,s) so that company h controls company s. There are at most 100 companies.

Write a program to:

1. Read from an ASCII input file, COMPANY.DAT, the list of triples, (i,j,p), to be considered for each case (that is, each data set), where i, j and p are positive integers. Different cases (data sets) will be separated with a blank record.
2. Find all the pairs (h,s) so that company h controls company s.
3. Write to an ASCII output file, COMPANY.SOL, all the pairs (h,s) found, with h different from s. The pairs (h,s) must be written in consecutive records and in increasing order of h. The solutions for different cases must be separated with a blank record.

Παράδειγμα:

COMPANY.DAT			COMPANY.SOL	
2	3	25	4	2
1	4	36	4	3
4	5	63	4	5
2	1	48		
3	4	30		
4	2	52		
5	3	30		
1	2	30	2	3
2	3	52	2	4
3	4	51	2	5
4	5	70	3	4
5	4	20	3	5
4	3	20	4	5

Πρόβλημα 3. Παραλληλόγραμμα στο χαρτί

N rectangles of different colours are superposed on a white sheet of paper. The sheet's sizes are: a cm wide and b cm long. The rectangles are put with their sides parallel to the sheet's borders. All rectangles fall within the borders of the sheet. As result, different figures of different colours will be seen. Two regions of the same colour are considered to be part of the same figure if they have at least one point in common; otherwise, they are considered different figures. The problem is to calculate the area of each of these figures. a, b are even positive integers not greater than 30.

The coordinate system considered has origin at the sheet's center and the axes parallel to the sheet's borders:

Different data sets are written in an ASCII input file, RECTANG.DAT:

a, b and N will be in the first line of each data set, separated by a blank space.

In each of the next N lines you will find:

the integer coordinates of the position where the left lower vertex of the rectangle was put, followed by the integer coordinates of the position where the upper right vertex of the rectangle was put and, then, the rectangle's colour represented by an integer between 1 and 64. White colour will be represented by 1. The order of the records corresponds to the order used to put the rectangles on the sheet. Different data sets will be separated with a blank record.

Write a program to:

1. Read the next data set from RECTANG.DAT
2. Calculate the area of each coloured figure
3. Write in an ASCII output file, RECTANG.SOL, the colour and the area of each coloured figure as shown in the example below. These records will be written in increasing order of colour. The solutions to different data sets will be separated by a blank record.

Example:

RECTANG.DAT	RECTANG.SOL
20 12 5	1 172
7 -5 -3 -1 4	2 47
5 -3 5 3 2	4 12
4 -2 -2 2 4	4 8 2 -2 3 -1 12
	12 1 3 1 7 5 1
30 30 2	1 630
0 0 5 14 2	2 70
10 -7 0 13 15	15 200

Πρόβλημα 4. Διαγωνισμός αεροπορικής εταιρείας

You have won a contest sponsored by an airline. The prize is a ticket to travel around Canada, beginning in the most western point served by this airline, then traveling only from west to east until you reach the most eastern point served, and then coming back only from east to west until you reach the starting city. No city may be visited more than once, except for the starting city, which must be visited exactly twice (at the beginning and the end of the trip). You are not allowed to use any other airline or any other means of transportation. Given a list of cities served by the airline, and a list of direct flights between pairs of cities, find an itinerary which visits as many cities as possible and satisfies the above conditions beginning with the first city and visiting the last city on the list and returning to the first city.

Different data sets are written in an ASCII input file, C:\IOI\ITIN.DAT. Each data set consists of:

- in the first line: the number N of cities served by the airline and the number V of direct flights that will be listed. N will be a positive integer not larger than 100. V is any positive integer.
- in each of the next N lines: a name of a city served by the airline. The names are ordered from west to east in the input file. That is, the i-th city is west of the j-th city if and only if $i < j$ (There are no two cities in the same meridian). The name of each city is a string of, at most, 15 digits and/or characters of the Latin alphabet, for example: AGR34 or BEL4 (There are no spaces in the name of a city)
- in each of the next V lines: two names of cities, taken from the list of cities, separated by a blank space. If the pair city1 city2 appears in a line, it indicates that there exists a direct flight from city1 to city2 and also a direct flight from city2 to city1.

Different data sets will be separated by an empty record (that is, a line containing only the end of line character). There will be no empty record after the last data set. The following example is stored in file C:\IOI\ITIN.DAT.

```

8 9                      5 5
Vancouver                C1
Yellowknife              C2
Edmonton                 C3
Calgary                  C4
Winnipeg                 C5
Toronto                  C5 C4
Montreal                 C2 C3
Halifax                  C3 C1
Vancouver Edmonton      C4 C1
Vancouver Calgary        C5 C2
Calgary Winnipeg
Winnipeg Toronto
Toronto Halifax
Montreal Halifax
Edmonton Montreal
Edmonton Yellowknife
Edmonton Calgary

```

The input may be assumed correct. No checking is necessary.

The solution found for each data set must be written to an ASCII output file, C:\IOI\ITIN.SOL: in the first line, the total number of cities in the input data set; in the next line, the number M of different cities visited in the itinerary, and in the next M+1 lines the names of the cities, one per line, in the order in which they are visited. Note the first city visited must be the same as the last. Only one solution is required. If no solution is found for a data set, only two records for this data set must be written in ITIN.SOL, the first one giving the total number of cities, and the second one saying: "NO SOLUTION".

A possible solution for the above example:

```

ITIN.SOL
8                      5
7                      NO SOLUTION
Vancouver
Edmonton
Montreal
Halifax
Toronto
Winnipeg
Calgary
Vancouver

```

6η Διεθνής Ολυμπιάδα Πληροφορικής 1994

Η έκτη ολυμπιάδα πληροφορικής έγινε στο Χάνινγκε της Σουηδίας.

Ημέρα 1η

Πρόβλημα 1.

Στην εικόνα 1 φαίνεται ένα τρίγωνο. Γράψτε ένα πρόγραμμα το οποίο να υπολογίζει το μεγαλύτερο άθροισμα αριθμών που υπάρχει πάνω σε μία διαδρομή από την κορυφή του δέντρου μέχρι τη βάση του.

```
      7
     3 8
    8 1 0
   2 7 4 4
  4 5 2 6 5
Εικόνα 1
```

- Σε κάθε βήμα μετακινούμαστε διαγώνια προς τα κάτω είτε δεξιά είτε αριστερά.
- Ο αριθμός των γραμμών στο τρίγωνο είναι > 1 και ≤ 100 .
- Οι αριθμοί στο τρίγωνο είναι όλοι ακέραιοι από 0..99.

Στο παραπάνω παράδειγμα η διαδρομή μέσω των 7,3,8,7,5 παράγει το μεγαλύτερο άθροισμα που είναι 30.

ΔΕΔΟΜΕΝΑ ΕΙΣΟΔΟΥ

Τα δεδομένα διαβάζονται από το αρχείο INPUT.TXT. Στην αρχή υπάρχει ο αριθμός των γραμμών του τριγώνου και μετά τα δεδομένα κάθε γραμμής.

Στο παράδειγμα μας, το INPUT.TXT είναι το παρακάτω :

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

ΔΕΔΟΜΕΝΑ ΕΞΟΔΟΥ

Το μεγαλύτερο άθροισμα γράφεται σαν ακέραιος στο αρχείο OUTPUT.TXT. Στο παράδειγμά μας το αρχείο αυτό θα έπρεπε να περιέχει τον αριθμό 30.

Πρόβλημα 2. Κάτοψη Κάστρου

```
N
W + E
S
```

Στην εικόνα 1 φαίνεται ο χάρτης ενός κάστρου. Γράψτε ένα πρόγραμμα που υπολογίζει :

- Τον αριθμό των δωματίων του κάστρου.
- Το μέγεθος (σε κομμάτια) του μεγαλύτερου δωματίου.
- Ποιόν τοίχο (που ενώνει δύο δωμάτια) να αφαιρέσουμε από το κάστρο, ώστε να κάνουμε ένα δωμάτιο όσο μεγαλύτερο γίνεται.

```
      1  2  3  4  5  6  7
*****
1*      *      *      *
****      *      *      *
2* *      *      *      *
* *****      *      *
3*      *      *      *
* *****      *      *
4* *      *      *
*****
Εικόνα 1. (Ένα string από (*) είναι
τοίχος)
```

Το κάστρο διαιρείται σε έναν πίνακα από τετράγωνα κομμάτια m γραμμών και n στηλών ($m \leq 50$, $n \leq 50$). Κάθε κομμάτι έχει από 1 μέχρι 4 τοίχους.

ΔΕΔΟΜΕΝΑ ΕΙΣΟΔΟΥ

Ο χάρτης βρίσκεται στο αρχείο INPUT.TXT στη μορφή αριθμών, έναν για κάθε κομμάτι.

- Το αρχείο ξεκινά με τον αριθμό των κομματιών στην κατεύθυνση βορρά - νότου και τον αριθμό των κομματιών στην κατεύθυνση ανατολής - δύσης.
- Στις επόμενες γραμμές του αρχείου κάθε κομμάτι περιγράφεται από έναν αριθμό ($0 \leq p \leq 15$). Ο αριθμός αυτός είναι το άθροισμα των : 1 (= υπάρχει τριόχος προς τη δύση), 2 (= υπάρχει τοίχος προς το βορρά), 4 (= τοίχος προς την ανατολή), 8 (= τοίχος προς το νότο). Οι εσωτερικοί τοίχοι ορίζονται δύο φορές. Ένας τοίχος προς τον νότο για το κομμάτι 1,1 υπάρχει επίσης και σαν τοίχος προς τον βορρά για το κομμάτι 2,1. Το κομμάτι στη θέση 1,1 έχει έναν τοίχο δυτικά, βόρρεια και νότια που σημαίνει ότι έχει άθροισμα $1+2+8 = 11$.
- Το κάστρο έχει πάντα τουλάχιστον δύο δωμάτια.

Το INPUT.TXT για το παράδειγμα μας:

```
4
7
11 6 11 6 3 10 6
7 9 6 13 5 15 5
```

```

1   10 12 7 13 7 5
13 11 10 8 10 12 13

```

ΔΕΔΟΜΕΝΑ ΕΞΟΔΟΥ

Στο αρχείο ΟΥΤΡ UT.TXT, γράφονται τα παρακάτω σε τρεις γραμμές:

- Ο αριθμός των δωματίων.
- Το εμβαδόν του μεγαλύτερου δωματίου σε κομμάτια.
- Μία πρόταση για τον τοίχο που πρέπει να αφαιρεθεί και θα ενώσει δύο δωμάτια, έτσι ώστε να φτιάξει το μεγαλύτερο δυνατό δωμάτιο (πρώτα τη γραμμή και μετά τη στήλη από το κομμάτι δίπλα στον τοίχο και στο τέλος την κατεύθυνση προς την οποία είναι ο τοίχος. Μπορεί να υπάρχουν περισσότερα κομμάτια, αλλά διαλέξτε ένα για να το τυπώσετε.

(«4 1 E» είναι π.χ. μία από πολλές πιθανές λύσεις)

```

5
9
4 1 E

```

Η τελευταία γραμμή αναφέρεται στην αφαίρεση του τοίχου που φαίνεται παρακάτω :

```

      1   2   3   4   5   6   7
*****
1*          *          *          *
****          *          ****
2* *          *          *          *
*   ****          ****          *
3*          *          *          *
*   ****          ****          *
4*->*          *          *
*****

```

Fig 2.(αφαίρεσε τον τοίχο προς τα ανατολικά του κομματιού 4 1)

Πρόβλημα 3. Μαγικά τετράγωνα με 5ψήφιους πρώτους

Στην εικόνα 1 φαίνεται ένα τετράγωνο. Κάθε γραμμή, κάθε στήλη και οι δύο διαγώνιες μπορεί να διαβαστεί σαν ένας 5ψήφιος πρώτος αριθμός. Οι γραμμές διαβάζονται από αριστερά προς τα δεξιά, οι στήλες από πάνω προς τα κάτω και οι δύο διαγώνιες από αριστερά προς τα δεξιά.

Χρησιμοποιώντας τα δεδομένα στο αρχείο INPUT.TXT, γράψτε ένα πρόγραμμα που να φτιάχνει τέτοια τετράγωνα.

- Οι πρώτοι αριθμοί πρέπει να έχουν το ίδιο άθροισμα ψηφίων (11 στην εικόνα 1).
- Το ψηφίο στην πάνω - αριστερή γωνία του τετραγώνου είναι προκαθορισμένο (1 στην εικόνα 1).
- Ένας πρώτος αριθμός μπορεί να χρησιμοποιηθεί πολλές φορές στο ίδιο τετράγωνο.
- Αν υπάρχουν κι άλλες λύσεις, τότε όλες πρέπει να δειχθούν.
- Ένας πενταψήφιος πρώτος αριθμός δεν μπορεί να ξεκινά με μηδενικό, δηλαδή το 00003 δεν είναι πενταψήφιος πρώτος αριθμός.

1	1	3	5	1
3	3	2	0	3
3	0	3	2	3
1	4	0	3	3
3	3	3	1	1

εικόνα 1

ΔΕΔΟΜΕΝΑ ΕΙΣΟΔΟΥ

Το πρόγραμμα διαβάζει τα δεδομένα από το αρχείο INPUT.TXT. Πρώτα είναι το άθροισμα των ψηφίων των πρώτων αριθμών και μετά το ψηφίο στην επάνω αριστερή γωνία του τετραγώνου. Το αρχείο περιέχει δύο γραμμές. Πάντα υπάρχει λύση με τα δεδομένα που δίδονται. Στο παράδειγμά μας είναι :

```

11
1

```

ΔΕΔΟΜΕΝΑ ΕΞΟΔΟΥ

Στο αρχείο ΟΥΤΡUT.TXT, γράψτε πέντε γραμμές για κάθε λύση που βρήκατε, σε κάθε γραμμή έναν πενταψήφιο πρώτο. Το παραπάνω παράδειγμα έχει τρεις λύσεις που σημαίνει ότι το ΟΥΤΡUT.TXT περιέχει τα ακόλουθα (η κενή γραμμή μεταξύ των λύσεων είναι προαιρετική) :

```

1 1 3 5 1
1 4 0 3 3
3 0 3 2 3
5 3 2 0 1
1 3 3 1 3

1 1 3 5 1
3 3 2 0 3
3 0 3 2 3
1 4 0 3 3
3 3 3 1 1

1 3 3 1 3
1 3 0 4 3
3 2 3 0 3

```

```

5 0 2 3 1
1 3 3 3 1

```

Ημέρα 2η

Πρόβλημα 1.

The nine numbers in figure 1 represent the position of 9 dials where each dial has one of four positions North or 12 o'clock (0), East or 3 o'clock (1), South or 6 o'clock (2) and West or 9 o'clock (3).

```

3 3 0
2 2 2
2 1 2
Figure 1 (Dial Positions)

```

```

0
3-+-1
2

```

There are 9 different ways to turn the dials on the clocks. Each way is called a move. Each move is selected by a number from 1 to 9. That number will turn the dials marked by a 1 90 degrees clockwise. Those marked with a 0 have no affect. The nine moves are displayed in figure 2 below.

```

1 1 0      1 1 1      0 1 1
1 1 0      0 0 0      0 1 1
0 0 0      0 0 0      0 0 0
Move 1      Move 2      Move 3

1 0 0      0 1 0      0 0 1
1 0 0      1 1 1      0 0 1
1 0 0      0 1 0      0 0 1
Move 4      Move 5      Move 6

0 0 0      0 0 0      0 0 0
1 1 0      0 0 0      0 1 1
1 1 0      1 1 1      0 1 1
Move 7      Move 8      Move 9

```

For example the following sequence of moves has the corresponding affect on the dials (they all end up at 12 o'clock (0)).

```

3 3 0      3 0 0
2 2 2      3 3 3      Move 8 ->
2 1 2      2 2 2

3 0 0      0 0 0
3 3 3      0 3 3      Move 9 ->
3 3 3      0 3 3

0 0 0
0 0 0
0 0 0

```

The problem is write a program that will take any starting position for the clock and find the shortest sequence of moves which puts all the dials in the 12 o'clock (0) position.

INPUT DATA

Read nine numbers from the INPUT.TXT file. The example above will have the input data file:

```

3 3 0
2 2 2
2 1 2

```

OUTPUT DATA

Write to the OUTPUT.TXT file the shortest sequence of moves (numbers), which turns all the dials to the 0 or 12 o'clock position. In our example the OUTPUT.TXT file could look as follows:

5849

Only one solution is required.

Πρόβλημα 2. Σταθμός Λεωφορείων

A man arrives at a bus stop at 12:00. He remains there during 12:00 12:59. The bus stop is used by a number of bus routes. The man notes the times of arriving buses. The times when buses arrive are given with the following rules:

- 1) Buses on the same route arrive at regular intervals from 12:00 to 12:59.
- 2) Times are given in whole minutes from 1 to 59.
- 3) Each bus route has at least 2 buses arriving at the station between 12 and 12:59.

- 4) The number of bus routes in the test example will be ≤ 17 .
- 5) Buses from different routes may arrive at the same time.
- 6) Several bus routes can have the same time of first arrival and/or time interval. If two bus routes have the same starting time and interval, they are distinct and are both to be presented.

Find the fewest number of bus routes that must stop at the bus stop to satisfy the input data. For each bus route, output the starting time and the interval.

INPUT DATA

The input file, INPUT.TXT, contains a number n ($n \leq 300$) telling how many arriving buses have been noted, followed by the arrival times in ascending order.

Out example:

```
17
0 3 5 13 13 15 21 26 27 29 37 39 39 45 51 52 53
```

If two buses arrive at the same time, that time is listed twice.

OUTPUT DATA

Write a table to the OUTPUT.TXT file with one line for each bus route. Each line in the file give the time of arrival for the first bus and the time interval in minutes. The order of the bus routes does not matter. If there are several solutions, only one is required.

Our example gives:

```
0 13
3 12
5 8
```

Πρόβλημα 3.

Consider the magic list of 5 numbers

1 3 10 2 5

Any two number that are next to each other are considered neighbors. Also the two end numbers 1 and 5 as neighbors as if the list formed a circle of numbers. Starting with 2, we can form an unbroken sequence of integers from 2 to 21 using a single number in the list or by adding neighbors. Here is how the sequence is formed:

```
2, 3, 1+3=4, 5, 5+1=6, 2+5=7, 2+5+1=8, 5+1+3=9, 10, 2+5+1+3=11,
10+2=12, 3+10=13, 1+3+10=14,
3+10+2=15, 1+3+10+2=16, 10+5+2=17, 10+2+5+1=18,
5+1+3+10=19, 3+10+2+5=20, 1+3+10+2+5=21
```

You were given three number (n , m , and k) where

n = the length of the list of numbers

m = the starting number

k = (smallest possible value for a member of the list, i.e. all numbers must be greater or equal to k).

Your task is to choose n integers for the magic list where an unbroken sequence of all integers m , $m+1$, $m+2$, ..., max can be generated where max is as large as possible.

INPUT DATA

The INPUT.TXT file contains three integers (n, m, k). For the example the file would be:

```
5
2
1
```

OUTPUT DATA

The file OUTPUT.TXT must contain:

- 1) The highest number (max) that can be generated with the list of numbers.
- 2) All arrangements of numbers in a circle that produce a sequence from m to max . (One per line.) Each arrangement is a list of numbers starting with the smallest number (which is not necessarily unique.)

Note: 2 10 3 1 5 is not a valid solution, since it does not start with the smallest number.

(1 3 10 2 5) and (1 5 2 10 3) must both be included in the output.

Note that (1 1 2 3), (1 3 2 1), (1 2 3 1) and (1 1 3 2) should all be output.

The output for our example would be:

1	3	10	2	5
1	5	2	10	3
2	4	9	3	5
2	5	3	9	4

7η Διεθνής Ολυμπιάδα Πληροφορικής 1995

Η έβδομη ολυμπιάδα πληροφορικής έγινε στο Αϊντχόβεν της Ολλανδίας, από την 26η Ιουνίου μέχρι την 3η Ιουλίου 1995.

Ημέρα 1η

Πρόβλημα 1. Packing Rectangles

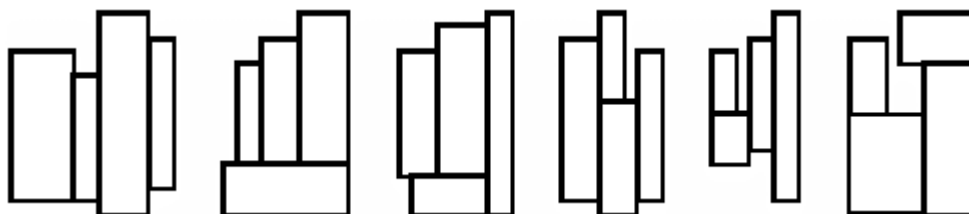


Figure 1: The six basic layouts of four rectangles

Four rectangles are given. Find the smallest enclosing (new) rectangle into which these four may be fitted without overlapping. By smallest rectangle we mean the one with the smallest area.

All four rectangles should have their sides parallel to the corresponding sides of the enclosing rectangle.

Figure 1 shows six ways to fit four rectangles together. These six are the only possible basic layouts, since any other layout can be obtained from a basic layout by rotation or reflection.

There may exist several different enclosing rectangles fulfilling the requirements, all with the same area.

You have to produce all such enclosing rectangles.

Input Data

The input file INPUT.TXT consists of four lines. Each line describes one given rectangle by two positive integers: the lengths of the sides of the rectangle. Each side of a rectangle is at least 1 and at most 50.

Output Data

The output file OUTPUT.TXT should contain one line more than the number of solutions. The first line contains a single integer: the minimum area of the enclosing rectangles (Subtask A). Each of the following lines contains one solution described by two numbers p and q with $p \leq q$ (Subtask B). These lines must be sorted in ascending order of p , and must all be different.

Example Input and Output

INPUT.TXT	OUTPUT.TXT
1 2	40
2 3	4 10
3 4	5 8
4 5	

Πρόβλημα 2. Shopping Offers

$=2$, $=5$, $++=5$, $++=10$, $++++=?$

In a shop each kind of product has a price. For example, the price of a flower is 2 ICU (Informatics Currency Units) and the price of a vase is 5 ICU. In order to attract more customers, the shop introduces some special offers.

A special offer consists of one or more product items for a reduced price. Examples: three flowers for 5 ICU instead of 6, or two vases together with one flower for 10 ICU instead of 12.

Write a program that calculates the price a customer has to pay for certain items, making optimal use of the special offers. That is, the price should be as low as possible. You are not allowed to add items, even if that would lower the price.

For the prices and offers given above, the (lowest) price for three flowers and two vases is 14 ICU: two vases and one flower for the reduced price of 10 ICU and two flowers for the regular price of 4 ICU.

Input Data

The input data appears in two files: INPUT.TXT and OFFER.TXT. The first file describes the purchases (in the 'shopping basket'). The second file describes the special offers. In both files, only integers are used.

The first line of INPUT.TXT contains the number b of different kinds of products in the basket ($0 \leq b \leq 5$).

Each of the next b lines contains three values c , k , and p . The value c is the (unique) product code ($1 \leq c \leq 999$). The value k indicates how many items of this product are in the basket ($1 \leq k \leq 5$). The value p is the regular price per item ($1 \leq p \leq 999$). Notice that all together at most $5 \cdot 5 = 25$ items can be in the basket.

The first line of OFFER.TXT contains the number s of special offers ($0 \leq s \leq 99$). Each of the next s lines describes one offer by giving its structure and its reduced price. The first number n on such a line is the number of different kinds of products that are part of the offer ($1 \leq n \leq 5$). The next n pairs of numbers (c, k) indicate that k items ($1 \leq k \leq 5$) with product code c ($1 \leq c \leq 999$) are involved in the offer. The last number p on the line stands for the reduced price ($1 \leq p \leq 9999$). The reduced price of an offer is less than the sum of the regular prices.

Output Data

Write to the output file OUTPUT.TXT one line with the lowest possible price to be paid for the purchases in the input file.

Example Input and Output

The product code of a flower is 7 and that of a vase is 8.

INPUT.TXT	OFFER.TXT	OUTPUT.TXT
2	2	14
7 3 2	1 7 3 5	
8 2 5	2 7 1 8 2 10	

Πρόβλημα 3. Printing

Note: This page is converted with a scanner, so it might contain small errors.

Client and Server

There are two users, each having a computer. The computers are identified by their names: CLIENT(1) and CLIENT(2). The two computers share one or more printers named SERVER(1), SERVER(2) and so on. The print jobs from both computers can only be executed in succession. To coordinate the communication between both computers with a printer we will use a special object: a semaphore.

Semaphore

Each printer has one related semaphore. A semaphore is in one of two states: S1 or S2. When the printer is free to receive a print job, the semaphore is in state S1. As long as the printer is busy with executing a print job the semaphore is in state S2. A semaphore can make two kinds of state transitions: 'S1→S2' and 'S2→S1'. When a user sends a print job to the computer, the computer will send a message "Are_you_open?" to the semaphore. If the state of the semaphore is S1, then the state of the semaphore will change to S2 and the semaphore will send a message "Open" to the computer that sent the message "Are_you_open?". If the state of the semaphore is S2 then the semaphore will send back a message: "Closed". After finishing a print job, the printer will send a message "Ready" to the semaphore. Upon receiving a message "Ready", the semaphore will change its state to S1.

Object type SEMAPHORE

In Documentation 1 you will find the specification of the object type SEMAPHORE. The specification includes the possible identifiers and the states of an object of the type SEMAPHORE, the 'Priority List', the 'Communication Diagram', the 'State Transition Diagram', and the 'Receive Procedures' for the messages:

"Ready" and "Are_you_open?". The 'Receive Procedures' describe how a semaphore responds to a message. The 'Priority List' is necessary because messages that are received at the same time by a semaphore can only be handled in succession. The 'Priority List' in Documentation 1 indicates that each message of a SERVER has a higher priority than a message of a CLIENT and that for instance a message of SERVER(2) has a higher priority than a message of SERVER(3).

Object type CLIENT

In Documentation 2 you will find the specification of the object type CLIENT. An object of the type CLIENT is in one of three states: SA, SB or SC. A client is in state SA when this client did not send a print job to a server and the servers are not busy with a print job of this client. A client which is in state SB, wants to have access to a server. The client can only get access to a server via a semaphore. A client is in state SC when a server is executing a print job of this client. A client can make three kinds of state transitions: 'SA→SB', 'SB→SC', 'SC→SA'. When a client is in state SB, the client can receive a message "Closed" from the semaphore. After receiving this message the client waits during a period, the 'Waiting_Period', before the client will again send a message "Are_you_open?" to the semaphore. When the semaphore sends a message "Open" to the client, this client changes its state to SC and sends the print job with a message "S_Job" to the server related to the sending semaphore. This server then executes the print job. After finishing the print job, the server will send two messages at the same time, a message "Ready" to its semaphore and a message "C_Ready" to the client. This client then changes its state from SC to SA. You can assume that the printers are ideal printers. They will finish each received print job. For instance there is never the situation 'out of paper'.

Communication

In Documentation 3 you find in the 'Communication Structure Diagram' all the message types that can be sent between the different object types. In the 'Message List' you find the specification of each message type. Each message has an identifier, a sender, a receiver, and sometimes a content. When a sender sends a message with the identifier A at time t , then the receiver will at time $t+1$ process the message by executing the 'Receive Procedure' A. If several senders send messages to the same receiver at time t , then the receiver will process all those messages at time $t+1$, in the same order in which the senders of these messages appear in the receiver's 'Priority List'.

Subtask A

A Local Area Network (LAN) includes the following objects at time 0:

Object: CLIENT(1), Client.State = SA, Waiting_Period = 2, Number_of_Servers= 1
Object: CLIENT(2), Client.State = SA, Waiting_Period = 1, Number_of_Servers= 1
Object: SERVER(1)
Object: SEMAPHORE(1), Semaphore.State = S1
In this LAN there have been sent, among others, the following messages:
At time 1:
CLIENT(1) sends a message with the identifier "Are_you_open?"
At time 2:
CLIENT(2) sends a message with the identifier "Are_you_open?"
At time 4:
SERVER(1) sends a message with the identifier "Ready"
At time 5:
CLIENT(1) sends a message with the identifier "Are_you_open?"

Documentation 4 indicates for SEMAPHORE(1), CLIENT(1) and CLIENT(2) in a time table up to time 6, which messages these objects receive, which messages they send, and in which state they are or to which state they change.

Question A.1

What would have happened if CLIENT(1) receives a message "C_Job" at time 4? Write your answer in Documentation 5.

Question A.2

What would have happened if CLIENT(2) receives a message "C_Job" at time 4? Write your answer in Documentation 5.

Question A.3

Complete the timetable in Documentation 4 up to time 13 if the following happens:

At time 8:
SERVER(1) sends a message with the identifier "Ready".
At time 10:
CLIENT(1) receives a message with the identifier "C_Job".
At time 12:
SERVER(1) sends a message with the identifier "Ready".

Subtask B

The LAN is extended. It now includes two semaphore-server pairs: (SEMAPHORE(1), SEMAPHORE(2), SERVER (1), SERVER(2)). For each CLIENT the Number_of_Servers equals 2.
To use both printers a change of the definition of the object type CLIENT described in Documentation 2 is necessary.

In Documentation 6 you find the changed 'Receive Procedures': C_Job and Wait.

In Documentation 6 you also find a description of the situation at time 0.

At the following times the following messages are received:

At time 0:
CLIENT(1) receives a message "C_Job" of a user.
At time 0:
CLIENT(2) receives a message "C_Job" of a user.

At time 4:
SEMAPHORE(1) receives a message "Ready".

What happens in the extended LAN according to the changed object type CLIENT?

Mark the correct answer in Documentation 6.

Subtask C

The definition change of the object type CLIENT in subtask B was not efficient for an extended LAN with more than one semaphore-server pair.

Subtask C.1

Change the definition of the object type CLIENT (see Documentation 2) so that the LAN with more than one semaphore-server pair can function as follows:

An object CLIENT(1) may use any of the servers in the LAN, but CLIENT(i) can only have one print job executed at a time by the servers. Each print job of a client is printed only once.

An object CLIENT(i) sends a message "Are_you_open?" successively to the several semaphores until a certain threshold value of received messages "Closed" has been reached or until the object CLIENT(i) receives one message "Open".

When the object CLIENT(i) reaches the threshold value, the object CLIENT(i) waits during a certain Waiting_Period before it again sends a series of messages "Are_you_open?".

Write your solution in Documentation 7.

Subtask C.2

Change the object type CLIENT in such a way, that object CLIENT(i) now may have several print jobs executed at a time by several servers, too. However, the number of print jobs of each CLIENT(i) should be

limited to CLIENT(i).Job_Maximum.

Write your solution in Documentation 8.

Documentation 1

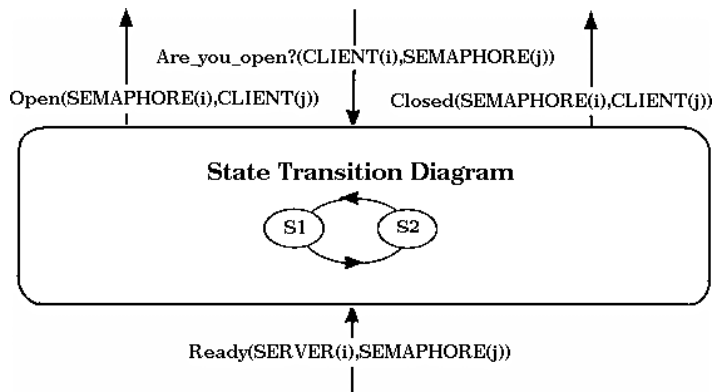
Object type: SEMAPHORE

Possible Identifiers: (SEMAPHORE(1),SEMAPHORE(2),SEMAPHORE(3),...)

State: (S1,S2); initial state is S1.

Priority List: SERVER(1),SERVER(2),...,CLIENT(1),CLIENT(2),...

Communication Diagram



Receive Procedures

```

procedure Are_you_open?(Client,Semaphore)
begin
if    State = S1
then  State <- S2
Send( "Open(Semaphore,Client)" )
else
if    State = S2
then  Send( "Closed(Semaphore,Client)" )
end

```

```

procedure Ready(Server, Semaphore)
begin
State <- S1
end

```

Documentation 2

Object type: CLIENT

Possible Identifiers: (CLIENT(1),CLIENT(2),CLIENT(3),...)

State: (SA,SB,SC); initial state is SA.

Priority List: CLIENT,SERVER(1),SERVER(2),...,SEMAPHORE(1),SEMAPHORE(2),...,USER(1),USER(2),...

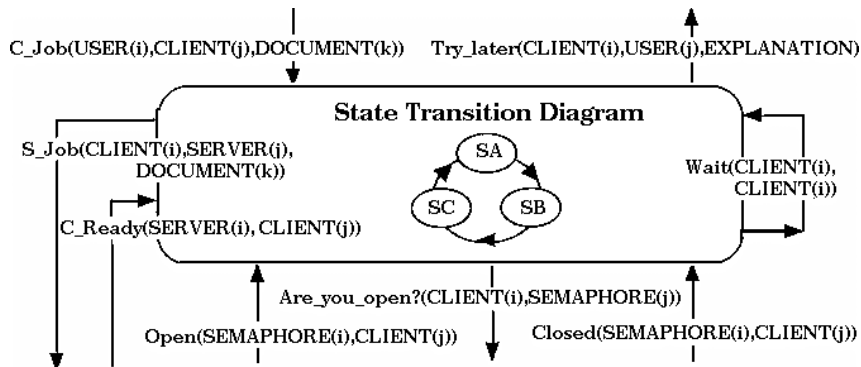
Countdown: { t | t in N }; initial value is 0.

Waiting_Period: { t | t in N and t > 0 }

Semaphore_Index: { i | i = 1,2,...,Number_of_Servers }

Number-of-Servers : { i | i in N and i > 0 }

Communication Diagram



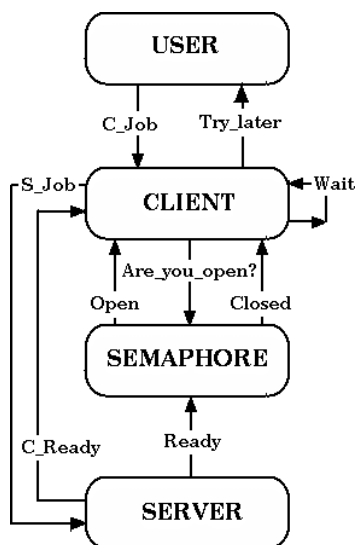
Receive Procedures

```

procedure C_Job(User,Client,Document)
begin
if    State = SA
then  State <- SB
Send( "Are_you_open?(Client,
SEMAPHORE(Semaphore_Index)) " )
else
if    State = SB
then  Send( "Try_later(Client,User,
Client_is_busy)" )
else
if    State = SC
then  Send( "Try_later(Client,User,
All_Servers_are_busy)" )
end
end
procedure Open(Semaphore,Client)
begin
if    State = SB
then  State <- SC
Send( "S_Job(Client,Server,

```

Documentation 3
Communication Structure Diagram



identifier	sender	receiver	content
Are_you_open?	CLIENT(i)	SEMAPHORE(j)	-
C_Job	USER(i)	CLIENT(j)	DOCUMENT(k)
C_Ready	SERVER(i)	CLIENT(j)	-
Closed	SEMAPHORE(i)	CLIENT(j)	-
Open	SEMAPHORE(i)	CLIENT(j)	-
Ready	SERVER(i)	SEMAPHORE(j)	-
S_Job	CLIENT(i)	SERVER(j)	DOCUMENT(k)
Try_later	CLIENT(i)	USER(j)	EXPLANATION
Wait	CLIENT(i)	CLIENT(i)	-

SEMAPHORE(1)				CLIENT(1)				CLIENT(2)			
received	State/	sent		received	State/	Count-	sent	received			
State/	Count-	sent									
messages	Transition	down	messages	messages	Transition	down	messages				
messages	Transition	down	messages								
time											
0		S1		SA		0					SA
0											

Question A.2

What would have happened if CLIENT(2) receives a message "C_Job" at time 4 ?

Documentation 6

Subtask B

The changed 'Receive Procedures': C_Job and Wait.

Note: The changed code is made italic in this page.

```
procedure C_Job(User,Client)
begin
  if      State = SA
  then    State <- SB
          for Semaphore_Index <- 1 step 1 until Number_of_Servers
            Send("Are_you_open?(Client,SEMAPHORE(Semaphore_Index))")
  else
    if      State = SB
    then    Send("Try_later(Client,User,Client_is_busy)")
  else
    if      State = SC
    then    Send(",Try_later(Client,User,All_Servers_are_busy)")
  end
procedure Wait(Client,Client)
begin
  if      State = SB
  then    Countdown <- Countdown -1
          if      Countdown > 0
          then    Send("Wait(Client,Client)")
  else
    if      Countdown < 0
    then    Countdown <- 0
    else    for Semaphore_Index <- 1 step 1 until Number_of_Servers
            Send("Are_you_open?(Client,SEMAPHORE(Semaphore_Index))")
  end
At time 0 the situation in the LAN is as follows:
Object: CLIENT(1), Client.State SA, Waiting_Period = 2, Number_of_Servers=2
Object: CLIENT(2), Client.State SA, Waiting_Period = 1, Number_of_Servers=2
Object: SEMAPHORE(1), Semaphore.State = S1
Object: SEMAPHORE(2), Semaphore.State = S1
At the following times the following messages are received:
At time 0: CLIENT(1) receives a message "C_Job" of a user.
At time 0: CLIENT(2) receives a message "C_Job" of a user.
At time 4: SEMAPHORE(1) receives a message "Ready".
What happens in the extended LAN according to the changed object type CLIENT?
```

Mark the correct answer.

- (a) The print job of CLIENT(1) will be printed on SERVER(1) and SERVER(2).
The print job of CLIENT(2) will not be printed.
- (b) The print job of CLIENT(1) will be printed once on SERVER(1).
The print job of CLIENT(2) will be printed once on SERVER(2).
- © The print job of CLIENT(1) will be printed once on SERVER(1).
The print job of CLIENT(2) will be printed once on SERVER(1).
- (d) The print job of CLIENT(1) will be printed once on SERVER(2).
The print job of CLIENT(2) will be printed once on SERVER(2).
- (e) The print job of CLIENT(1) will not be printed.
The printjob of CLIENT(2) will be printed on SERVER(1) and SERVER(2).

Documentation 7

Subtask C.1

Ημέρα 2η

Πρόβλημα 1. Παιχνίδι με Γράμματα

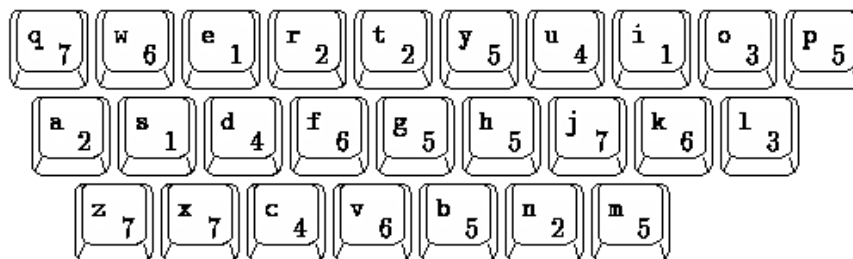


Figure 1: Each of the 26 lowercase letters and its value

Letter games are popular at home and on television. In one version of the game, every letter has a value, and you collect letters to form one or more words giving the highest possible score. Unless you have 'a way with words', you will try all the words you know, sometimes looking up the spelling, and then compute the scores. Obviously, this can be done more accurately by computer.

Given the values in Figure 1, a list of English words, and the letters collected: find the highest scoring words or pairs of words that can be formed.

Input Data

The input file INPUT.TXT contains one line with a string of lowercase letters (from 'a' to 'z'): the letters collected. The string consists of at least 3 and at most 7 letters in arbitrary order. The 'dictionary' file WORDS.TXT consists of at most 40,000 lines. At the end of this file is a line with a single period ('.'). Each of the other lines contains a string of at least 3 and at most 7 lowercase letters. The file WORDS.TXT is sorted alphabetically and contains no duplicates.

Output Data

On the first line of file OUTPUT.TXT, your program should write the highest score (Subtask A), and on each of the following lines, all the words and/or word pairs from file WORDS.TXT with this score (Subtask B). A letter must not occur more often in an output line than in the input line. Use the letter values given in Figure 1.

When a combination of two words can be formed with the given letters, the words should be printed on the same line separated by a space. Do not duplicate pairs; for example, 'rag prom' and 'prom rag' are the same pair, therefore only one of them should be written. A pair in an output line may consist of two identical words.

Example Input and Output

WORDS.TXT	INPUT.TXT	OUTPUT.TXT
profile	prmgroa	24
program		program
prom		prom rag
rag		
ram		
rom		
.		

Πρόβλημα 2. Αγώνας Δρόμου

Figure 1 gives an example of a course for a street race. You see some points, labeled from 0 to N (here N=9), and some arrows connecting them. Point 0 is the start of the race; point N is the finish. The arrows represent one-way streets. The participants of the race move from point to point via the streets, in the direction of the arrows only. At each point, a participant may choose any outgoing arrow.

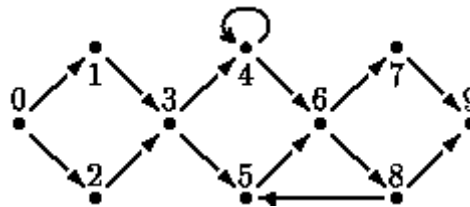


Figure 1: A street course with 10 points

A well-formed course has the following properties:

1. Every point in the course can be reached from the start.
2. The finish can be reached from each point in the course.
3. The finish has no outgoing arrows.

A participant does not have to visit every point of the course to reach the finish. Some points, however, are unavoidable. In the example, these are points 0, 3, 6, and 9. Given a well-formed course, your program has to determine the set of unavoidable points that all participants have to visit, excluding start and finish (Subtask A).

Suppose the race has to be held on two consecutive days. For that purpose the course has to be split into two courses, one for each day. On the first day, the start is at point 0, and the finish at some 'splitting point'. On the second day, the start is at this splitting point and the finish is at point N. Given a well-formed course, your program has to determine the set of splitting points (Subtask B). A point S is a splitting point for the well-formed course C if S differs from the start and the finish of C, and the course can be split into two well-formed courses that have no common arrows and that have S as only common point. In the example, only point 3 is a splitting point.

Input Data

The file INPUT.TXT describes a well-formed course with at most 50 points and at most 100 arrows. There are N+1 lines in the file. The first N lines contain the endpoints of the arrows that leave from the points 0 through N-1 respectively. Each of these lines ends with the number -2. The last line contains the number -1.

Output Data

Your program should write two lines to the file OUTPUT.TXT. The first line should contain the number of unavoidable points in the input course, followed by the labels of these points, in any order (Subtask A). The second line should contain the number of splitting points of the input course, followed by the labels of all these points, in any order (Subtask B).

Example Input and Output

INPUT.TXT	OUTPUT.TXT
1 2 -2	2 3 6
3 -2	1 3
3 -2	
5 4 -2	
6 4 -2	
6 -2	
7 8 -2	
9 -2	
5 9 -2	
-1	

Πρόβλημα 3. Καλώδια και Διακόπτες

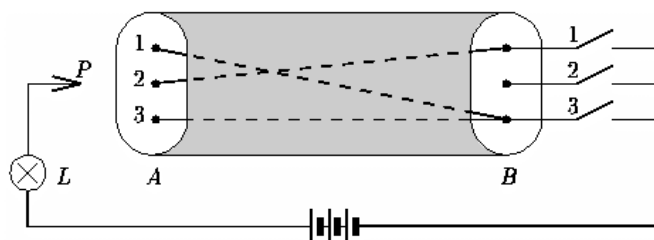


Figure 1: Cable with three wires and three switches

In Figure 1, a cable with three wires connects side A to side B. On side A, the three wires are labeled 1, 2, and 3. On side B, wires 1 and 3 are connected to switch 3, and wire 2 is connected to switch 1.

In general, the cable contains m wires ($1 \leq m \leq 90$), labeled 1 through m on side A, and there are m switches on side B, labeled 1 through m. Each wire is connected to exactly one of the switches. Each switch can be connected to zero or more wires.

Measurements

Your program has to determine how the wires are connected to the switches by doing some measurements.

Each switch can be made either conducting or non-conducting. Initially all switches are non-conducting. A wire can be tested on side A with probe P: Lamp L will light up if and only if the sensed wire is connected to a conducting switch.

Your program begins by reading one line with the number m from standard input. It then can give three kinds of commands by writing a line to `\emph{standard output}`. Each command starts with a single uppercase letter: T (Test a wire), C (Change a switch), and D (Done). Command T is followed by a wire label, C by a switch label, and D by a list whose i -th element is the label of the switch to which wire i is connected.

After commands T and C, your program should read one line from `\emph{standard input}`. Command T returns Y (Yes) when the wire's switch is conducting (the lamp lights up), otherwise it returns N (No).

Command C returns Y if the new switch state is conducting, and N otherwise. The effect of command C is to change the state of the switch (if it was conducting then it will be non-conducting afterwards and vice versa); the result is returned just for feedback.

Your program may give commands T and C mixed in any order. Finally, it gives command D and terminates.

Your program should give no more than nine hundred (900) commands in total.

Example

Figure 2 presents an example conversation involving 8 commands relating to Figure 1.

Standard Output	Standard Input
	3
C 3	Y
T 1	Y
T 2	N
T 3	Y
C 3	N
C 2	Y
T 2	N
D 3 1 3	

Figure 2: Example conversation

8η Διεθνής Ολυμπιάδα Πληροφορικής 1996

Η όγδοη ολυμπιάδα πληροφορικής έγινε στο Βέτσπριμ της Ουγγαρίας.

Ημέρα 1η

Πρόβλημα 1. Το παιχνίδι

Consider the following two-player game. The game board consists of a sequence of positive integers. The two players move alternately. When a player moves, he selects a number from either the left or the right end of the sequence. The selected number is deleted from the board. The game is over when all numbers have been selected. The first player wins if the sum of the numbers he has selected is at least as much as selected by the second player. The second player plays his best.

The first player starts the game.

If the board initially contains an even number of elements, then the first player has a winning strategy. You are to write a program that implements the strategy of the first player to win the game. The second player's response is provided by a given computer program. The two players communicate with three procedures of the module Play that is made available to you. These procedures are StartGame, MyMove and YourMove. The first player should initiate the game by executing the parameterless procedure StartGame. If the first player selects a number from the left end, he executes the procedure MyMove('L'). Similarly, executing the instruction MyMove('R') sends a message to the second player indicating that the first player has selected a number from the right end. The second player, i.e. the machine moves immediately, and the first player can learn this move by executing the instruction YourMove©, where C is a character type variable (in C/C++ you write this as YourMove(&C)). The value of C is 'L' or 'R' depending on whether the selection has been made either from the left or the right end.

Input Data

The first line of file INPUT.TXT contains the size N of the initial board. N is even and $2 \leq N \leq 100$. The remaining N lines contain one number in each line, the contents of the initial board in left to right order. Each number is at most 200.

Output Data

When the game is over, your program should write the final result of the game to the file OUTPUT.TXT. The file contains two numbers in the first line. The first number is the sum of the numbers selected by the first player and the second number is the sum of the numbers selected by the second player. Your program must play a game and the output must correspond to the game played.

Example Input and Output

```
INPUT.TXT
6472952
OUTPUT.TXT
15 14
```

Πρόβλημα 2. Job Processing

Figure 1

A factory is running a production line. Two operations have to be performed on each job: first operation «A», then operation «B». There is a certain number of machines capable of performing each operation. Figure 1 shows the organisation of the production line that works as follows. A type «A» machine takes a job from the input container, performs operation «A» and puts the job into the intermediate container. A type «B» machine takes a job from the intermediate container, performs operation «B» and puts the job into the output container.

All machines can work in parallel and independently of each other, and the size of each container is unlimited. The machines have different performance characteristics, a given machine works with a given processing time.

Give the earliest time operation «A» can be completed for all N jobs provided that the jobs are available at time 0. (Subtask A). Also compute the minimal amount of time that is necessary to perform both operations on N jobs (Subtask B).

Input Data

The file INPUT.TXT contains positive integers in five lines. The first line contains N, the number of jobs ($1 \leq N \leq 1000$). On the second line, the number M1 of type «A» machines ($1 \leq M1 \leq 30$) is given. In the third line there are M1 integers, the job processing times of each type «A» machine. The fourth and the fifth line contain the number M2 of type «B» machines ($1 \leq M2 \leq 30$) and the job processing times of

each type «B» machine, respectively. The job processing time is measured in units of time, which includes the time needed for taking a job from a container before processing and putting it into a container after processing. Each processing time is at least 1 and at most 20.

Output Data

Your program should write two lines to file OUTPUT.TXT. The first line should contain one positive integer: the solution of subtask A. The second line should contain the solution of subtask B.

Example Input and Output

```
INPUT.TXT
5
2
1 1
3
3 1 4
OUTPUT.TXT
3
5
```

Πρόβλημα 3. Το δίκτυο των σχολείων

A number of schools are connected to a computer network. Agreements have been developed among those schools: each school maintains a list of schools to which it distributes software (the «receiving schools»). Note that if B is in the distribution list of school A, then A does not necessarily appear in the list of school B.

You are to write a program that computes the minimal number of schools that must receive a copy of the new software in order for the software to reach all schools in the network according to the agreement (Subtask A). As a further task, we want to ensure that by sending the copy of new software to an arbitrary school, this software will reach all schools in the network. To achieve this goal we may have to extend the lists of receivers by new members. Compute the minimal number of extensions that have to be made so that whatever school we send the new software to, it will reach all other schools (Subtask B). One extension means introducing one new member into the list of receivers of one school.

Input Data

The first line of file INPUT.TXT contains an integer N: the number of schools in the network ($2 \leq N \leq 100$). The schools are identified by the first N positive integers. Each of the next N lines describes a list of receivers. The line i+1 contains the identifiers of the receivers of school i. Each list ends with a 0. An empty list contains a 0 alone in the line.

Output Data

Your program should write two lines to the file OUTPUT.TXT. The first line should contain one positive integer: the solution of subtask A. The second line should contain the solution of subtask B.

Example Input and Output

```
INPUT.TXT
5
2 4 3 0
4 5 0
0
0
1 0
OUTPUT.TXT
1
2
```

Ημέρα 2η

Πρόβλημα 1. Ταξινομώντας μια τριπλέτα

Sorting is one of the most frequently done computational tasks. Consider the special sorting problem, where the records to be sorted have at most three different key values. This happens for instance when we sort medalists of a competition according to medal value, that is, gold medalists come first, followed by silver, and bronze medalists come last.

In this task the possible key values are the integers 1, 2 and 3. The required sorting order is non-decreasing. Sorting has to be accomplished by a sequence of exchange operations.

An exchange operation, defined by two position numbers p and q, exchanges the elements in positions p and q.

You are given a sequence of key values. Write a program that computes the minimal number of exchange operations that are necessary to make the sequence sorted. (Subtask A).

Moreover, construct a sequence of exchange operations for the respective sorting (Subtask B).

Input Data

The first line of file INPUT.TXT contains the number of records $N(1 \leq N \leq 1000)$. Each of the following N lines contains a key value.

Output Data

Write on the first line of file OUTPUT.TXT the minimal number L of exchange operations needed to make the sequence sorted (Subtask A). The following L lines give the respective sequence of the exchange operations in the order performed. Each line contains one exchange operation described by two numbers p and q , the positions of the two elements to be exchanged (Subtask B). Positions are denoted by the numbers from 1 to N .

Example Input and Output

INPUT.TXT	OUTPUT.TXT
9	4
2	1 3
2	4 7
1	9 2
3	5 9
3	
3	
2	
3	
1	

Πρόβλημα 2. Longest Prefix

The structure of some biological objects is represented by the sequence of their constituents. These constituents are denoted by uppercase letters. Biologists are interested in decomposing a long sequence into shorter ones. These short sequences are called primitives. We say that a sequence S can be composed from a given set of primitives P , if there are n primitives p_1, \dots, p_n in P such that the concatenation $p_1 \dots p_n$ of the primitives equals S . By the concatenation of primitives p_1, \dots, p_n we mean putting them together in that order without blanks. The same primitive can occur more than once in the concatenation and not necessarily all primitives are present. For instance the sequence ABABACABAAB can be composed from the set of primitives

$\{A, AB, BA, CA, BBC\}$.

The first K characters of S are the prefix of S with length K . Write a program which accepts as input a set of primitives P and a sequence of constituents T . The program must compute the length of the longest prefix, that can be composed from primitives in P .

Input Data

The input data appear in two files. The file INPUT.TXT describes the set of primitives P , while the file DATA.TXT contains the sequence T to be examined. The first line of INPUT.TXT contains N , the number of primitives in P ($1 \leq N \leq 100$). Each primitive is given in two consecutive lines. The first line contains the length L of the primitive ($1 \leq L \leq 20$). In the second line there is a string of uppercase letters (from 'A' to 'Z') of length L . The N primitives are all different. Each line of the file DATA.TXT contains one uppercase letter in the first position. This file ends with a line containing a single period ('.'). The length of the sequence is at least 1 and at most 500,000.

Output Data

Write into the first line of file OUTPUT.TXT the length of the longest prefix of T that can be composed from the set P .

Example Input and Output

INPUT.TXT	DATA.TXT	OUTPUT.TXT
5	A	11
1	B	
A	A	
2	B	
AB	A	
3	C	
BBC	A	
2	B	
CA	A	
2	A	

BA B
 C
 B

Πρόβλημα 3. Μαγικά τετράγωνα

Following the success of the magic cube, Mr. Rubik invented its planar version, called magic squares. This is a sheet composed of 8 equal-sized squares:

1	2	3	4
8	7	6	5

In this task we consider the version where each square has a different colour. Colours are denoted by the first 8 positive integers (see Figure 3). A sheet configuration is given by the sequence of colours obtained by reading the colours of the squares starting at the upper left corner and going in clockwise direction. For instance, the configuration of Figure 3 is given by the sequence (1,2,3,4,5,6,7,8). This configuration is the initial configuration.

Three basic transformations, identified by the letters 'A', 'B' and 'C', can be applied to a sheet:

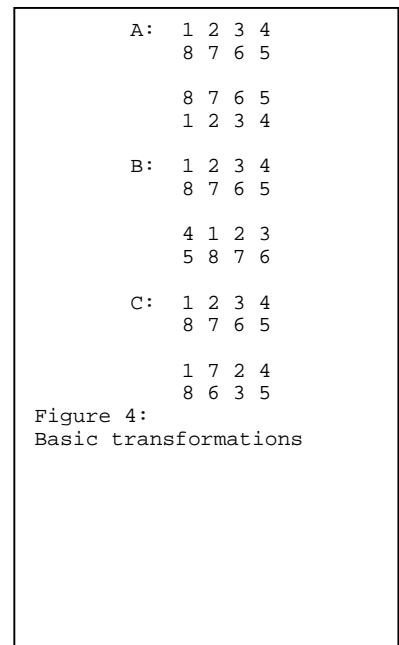
'A': exchange the top and bottom row,

'B': single right circular shifting of the rectangle,

'C': single clockwise rotation of the middle four squares.

All configurations are available using the three basic transformations.

The effects of the basic transformations are described in Figure 4. Numbers outside the squares denote square positions. If a square in position p contains number i, it means that after applying the transformation, the square whose position was i before the transformation moves to position p.



You are to write a program that computes a sequence of basic transformations that transforms the initial configuration of Figure 3 to a specific target configuration (Subtask A). Two extra points will be given for the solution if the length of the transformation sequence does not exceed 300 (Subtask B).

Input Data

The file INPUT.TXT contains 8 positive integers in the first line, the description of the target configuration.

Output Data

On the first line of file OUTPUT.TXT your program must write the length L of the transformation sequence. On the following L lines it must write the sequence of identifiers of basic transformations, one letter in the first position of each line.

Tool

MTOOL.EXE is a program in the task directory that lets you play with the magic squares. By executing «mtool input.txt output.txt» you can experiment with the target configuration and the sequence of transformations.

Example Input and Output

INPUT.TXT	OUTPUT.TXT
2 6 8 4 5 7 3 1	7
	B
	C
	A
	B
	C
	C
	B

9η Διεθνής Ολυμπιάδα Πληροφορικής 1997

Η ένατη ολυμπιάδα πληροφορικής έγινε στο Cape Town της Νότιας Αφρικής.

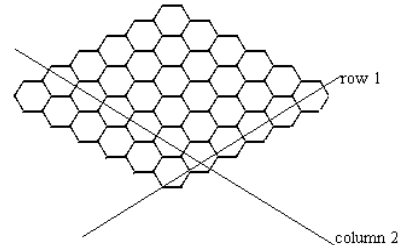
Ημέρα 1η

Πρόβλημα 1. The Game of Hex

The aim of the game is for the first player to connect a hex counter owned by her on column 1 to a hex counter owned by her on column N.

Rules of Hex:

Hex is a two player strategy game played on a $N \times N$ rhombus of hexagons, as illustrated here for $N=6$.



1. The two players of the game are your program and the evaluation library.
2. Your program always has the first move.
3. Players alternately place hex counters on the board.
4. A hex counter may be placed at any open position on the board.
5. Two hexagons are adjacent if they share an edge.
6. Hex counters on adjacent hexagons of the same player (contestant next to contestant, or evaluator next to evaluator) are *connected*.
7. Connectivity is transitive (and commutative): if hex1 is connected to hex2 and hex2 is connected to hex3 then hex3 is connected to hex1 and hex1 is connected to hex3.

Task:

- You are required to write a program which plays the game of Hex.
- The goal of the first player (your program) is to connect a hex counter of yours on column 1 to a hex counter of yours on column N.
- The other player (evaluator's program) attempts to connect an evaluator's hex counter on row 1 to an evaluator's hex counter on row N.
- If your program plays optimally, it will always win.

Input and Output:

Your program must **not** read from or write to any files. Your program must not receive keyboard input, and must **not** produce output on the screen. It will receive all its input from the functions in the hex library. The library will produce an output file named HEX.OUT; you should ignore its contents.

At the start of the game your program will be presented with a board that may have hex counters already placed, representing a state of a game such that the first player may still win. Your program must use the functions GetMax and LookAtBoard to determine the state of the board.

At the start of the game, an equal number of hexes belongs to the evaluation program and your program.

Constraints:

1. The size of the board will always be in the range **1** to **20** inclusive.
2. Your program may have to make up to 200 moves to complete a game. The entire game must be finished within 40 seconds. It is guaranteed that the evaluation library will complete its processing within 20 seconds.

Library:

A library called *HexLib* is provided which you must link to your code. An example file, for each programming language, showing how this is done is included in the task directory. These files are TESTHEX.CPP, TESTHEX.C, TESTHEX.PAS, and TESTHEX.BAS. If you are using QuickBasic you must include the library by typing

```
QB /L HEXLIB
```

The functions in HexLib are:

(in order of Pascal, C/C++ and Basic respectively)

```
function LookAtBoard (row, column: integer): integer;  
int LookAtBoard (int row, int column);  
declare function LookAtBoard cdecl (byval x as integer, byval y as integer)  
Returns
```


- 1 if row<1 or row>N or column<1 or column>N
- 0 if there is no hex counter at the position
- 1 if the hex counter at the specified position belongs to your program (player 1)
- 2 if the hex counter at the specified position belongs to the evaluation library (player 2)

```
procedure PutHex (row, column: integer);
void PutHex (int row, int column);
declare sub PutHex cdecl (byval x as integer, byval y as integer)
```

Places a contestant's hex counter at the specified row and column if the position is not occupied.

```
function GameIsOver: integer;
int GameIsOver (void);
declare function GameIsOver cdecl ()
```

Returns one of the following integers

- 0 the game is not over.
- 1 every position on the board is occupied by a hex counter.
- 2 your program has won.
- 3 the evaluation library has won.

```
procedure MakeLibMove;
void MakeLibMove(void);
declare sub MakeLibMove cdecl ()
```

Allows the evaluation library to calculate its next move and places its hex counter on the board. The change to the board will be indicated by *LookAtBoard* and the other functions.

```
function GetRow: integer;
int GetRow (void);
declare function GetRow cdecl ()
```

Returns the row of the hex counter placed by the evaluation library, or -1 if no hex counter has been placed yet. This function always returns the same value until your program calls *MakeLibMove* again.

```
function GetColumn: integer;
int GetColumn (void);
declare function GetColumn cdecl ()
```

Returns the column of the last hex counter placed by the evaluation library, or -1 if no hex counter has been placed yet. This function always returns the same value until your program calls *MakeLibMove* again.

```
function GetMax: integer;
int GetMax (void);
declare function GetMax cdecl ()
```

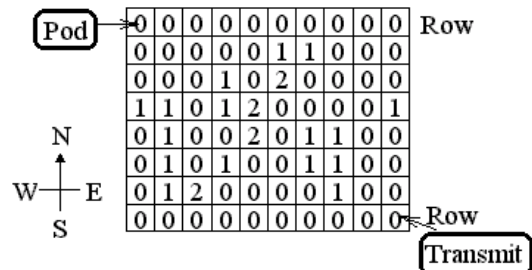
Returns the size of the board, N.

Scoring:

- If your program wins a game, it will score full marks for that data set.
- If your program loses a game, it will score 20% for that data set.
- If your program terminates before the end of a game or runs out of time, it will score 0 for that data set.

Πρόβλημα 2. Mars explorer

In a future mission to Mars, a pod, containing a number of Mars exploration vehicles (MEVs), will be landing on the surface of Mars. All the MEVs will be released the pod's landing site, from which they will move towards a transmitter that has landed a short distance away. While the vehicles move towards the transmitter, they are required to gather rock samples. A rock may only be sampled once, by the first MEV to visit the rock. After that the rock may not be sampled again, but other MEVs may pass the same position. The vehicles cannot move onto rough terrain. The design of the vehicle is such that it can only move south or east in a path that follows the grid pattern from the pod to the transmitter. More than one MEV may occupy the same position at the same time.



Warning: If a MEV gets stuck, its samples are lost and the sites sampled cannot be resampled.

Task:

Calculate the individual movement of the vehicles to maximise the number of rock samples collected and the number of MEVs that reach the transmitter, using the vehicles that landed with the pod.

Input:

The surface of the planet between the pod and the transmitter is represented by a grid P, Q with the pod position always at position (1,1) and the transmitter located at (P, Q). The definitions of the different types of terrain are as follows:

- Clear terrain: 0
- Rough terrain: 1
- Rock sample: 2

The input file consists of:

NumberOfVehicles

P

Q

(X_1Y_1)	(X_2Y_1)	$(X_3Y_1) \dots (X_{P-1}Y_1)$	(X_PY_1)
(X_1Y_2)	(X_2Y_2)	$(X_3Y_2) \dots (X_{P-1}Y_2)$	(X_PY_2)
(X_1Y_3)	(X_2Y_3)	$(X_3Y_3) \dots (X_{P-1}Y_3)$	(X_PY_3)
...			
(X_1Y_{Q-1})	(X_2Y_{Q-1})	$(X_3Y_{Q-1}) \dots (X_{P-1}Y_{Q-1})$	(X_PY_{Q-1})
(X_1Y_Q)	(X_2Y_Q)	$(X_3Y_Q) \dots (X_{P-1}Y_Q)$	(X_PY_Q)

P and Q are the size of the grid, and NumberOfVehicles is an integer less than 1000, representing the number of vehicles released by the pod. Q lines each represent a row in the surface representation. P and Q will not exceed 255.

Sample input:

MARS.DAT	Explanation:
2	Number of vehicles
10	Size of P
8	Size of Q
0 0 0 0 0 0 0 0 0 0	Row 1
0 0 0 0 0 1 1 0 0 0	Row 2
0 0 0 1 0 2 0 0 0 0	Row 3
1 1 0 1 2 0 0 0 0 1	Row 4
0 1 0 0 2 0 1 1 0 0	Row 5
0 1 0 1 0 0 1 1 0 0	Row 6
0 1 2 0 0 0 0 1 0 0	Row 7
0 0 0 0 0 0 0 0 0 0	Row 8

Output:

A sequence of lines representing the movements of the MEVs towards the transmitter. Each line contains a vehicle number and a digit 0 or 1, where 0 is a move South and 1 a move East.

Sample output:

MARS.OUT	Explanation:
1 1	vehicle 1 moves east
1 0	vehicle 1 moves south
2 1	vehicle 2 moves east
2 0	vehicle 2 moves south
1 1	...
1 1	
2 0	
2 1	
2 0	
2 0	
2 0	
2 0	
1 1	
1 0	
1 0	
1 0	
1 0	
1 0	
2 0	
2 1	
1 1	
1 1	
1 1	
1 1	
1 1	
2 1	
2 1	
2 1	
2 1	
2 1	
2 1	

Scoring:

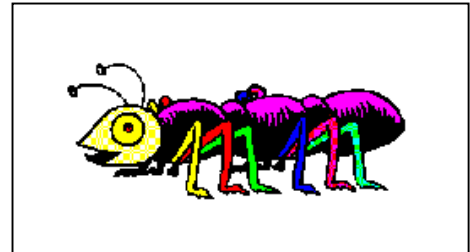
The calculation of the score will be based on the number of samples collected, as a proportion of the total possible samples, with adjustments made for the arrival or non-arrival of MEVs at the transmitter.

- An illegal move invalidates a solution set. An illegal move occurs when a MEV is moved over rough terrain, or outside the grid.
- $Score = (number\ of\ samples\ collected\ and\ taken\ to\ the\ transmitter + number\ of\ MEVs\ reaching\ the\ transmitter - number\ of\ MEVs\ not\ reaching\ the\ transmitter)$ as a % of the maximum possible score for the solution set.
- A maximum of 100% and a minimum of 0% can be scored.

Πρόβλημα 3. The Toxic iShongololo

"iShongololo" is the Zulu name for a millipede. They are long, shiny, black arthropods with many legs.

The iShongololo eats through an edible "fruit" which for the sake of this problem can be considered a rectangular solid with integer dimensions of L (length), W (width) and H (height).



Task:

You are required to write a program that maximizes the number of blocks eaten by the iShongololo without violating the constraints given. The program must output the actions that the iShongololo makes as it eats its way through the fruit.

The iShongololo starts outside the fruit. The first block the iShongololo must eat is 1, 1, 1 and it must then move to this block. It stops when no more blocks can be legally eaten and it can no longer move.

Constraints:

1. The iShongololo occupies exactly one empty block.
2. The iShongololo can only eat one complete block at a time.
3. The iShongololo cannot move to a position where it has previously moved to (that is, move backwards or cross its path).
4. The iShongololo cannot move to a solid (uneaten) block, or outside the fruit.
5. The iShongololo may only move to or eat blocks with whom it shares a face. It may only eat blocks which have no other faces exposed to empty eaten blocks.

Input:

As input your program will receive three numbers (integers) which are the length (L), width (W) and height (H) of the solid.

The three integers L, W, H, are each on a separate line. The three integers will be between 1 and 32 (inclusive).

Sample input:

TOXIC.DAT	Explanation:
2	Length of solid is 2.
3	Width of solid is 3.
2	Height of solid is 2.

Output:

The output consists of lines that begin with "E" (eat) or "M" (move) followed by 3 integers that represent the block eaten or moved to on the axes corresponding to L, W, H. For example the following is a valid solution for the input example.

Sample output (this may not be optimal):

TOXIC.OUT	Explanation:
E 1 1 1	Eat the block 1 1 1
M 1 1 1	Move to the block 1 1 1
E 2 1 1	Eat the block 2 1 1
E 1 1 2	Eat the block 1 1 2
E 1 2 1	Eat the block 1 2 1
M 1 2 1	Move to the block 1 2 1
E 1 3 1	Eat the block 1 3 1
M 1 3 1	Move to the block 1 3 1
E 2 3 1	Eat the block 2 3 1
E 1 3 2	Eat the block 1 3 2
M 1 3 2	Move to the block 1 3 2

Scoring:

- If the iShongololo violates the constraints, then your solution receives 0 points.
- The total score is the percentage of blocks eaten as a proportion of our best known solution.
- A solution cannot score more than 100%.

Πρόβλημα 1. Character Recognition

This problem requires you to write a program that performs character recognition.

Details:

Each ideal character image has 20 lines of 20 digits. Each digit is a '0' or a '1'. See Figure 1a for the layout of character images in the file.

The file FONT.DAT contains representations of 27 ideal character images in this order:

□abcdefghijklmnopqrstuvwxyz

where □ represents the space character.

The file IMAGE.DAT contains one or more potentially corrupted character images. A character image might be corrupted in these ways:

- at most one line might be duplicated (and the duplicate immediately follows)
- at most one line might be missing
- some '0's might be changed to '1's
- some '1's might be changed to '0's.

No character image will have both a duplicated line **and** a missing line. No more than 30% of the '0's and '1's will be changed in any character image in the evaluation datasets.

In the case of a duplicated line, one or both of the resulting lines may have corruptions, and the corruptions may be different.

Task:

Write a program to recognise the sequence of one or more characters in the image provided in file IMAGE.DAT using the font provided in file FONT.DAT.

Recognise a character image by choosing the font character images that require the smallest number of overall changed '1's and '0's to be corrupted to the given font image, given the most favourable assumptions about duplicated or omitted lines. Count corruptions in only the least corrupted line in the case of a duplicated line. All characters in the sample and evaluation images used are recognisable one-by-one by a well-written program. There is a unique best solution for each evaluation dataset.

A correct solution will use precisely all of the data supplied in the IMAGE.DAT input file.

Input:

Both input files begin with an integer N ($19 \leq N \leq 1200$) that specifies the number of lines that follow:

```
N
(digit1)(digit2)(digit3) ... (digit20)
(digit1)(digit2)(digit3) ... (digit20)
...
```

Each line of data is 20 digits wide. There are no spaces separating the zeros and ones.

The file FONT.DAT describes the font. FONT.DAT will always contain 541 lines. FONT.DAT may differ for each evaluation dataset.

Output:

Your program must produce a file IMAGE.OUT, which contains a single string of the characters recognised. Its format is a single line of ASCII text. The output should not contain any separator characters. If your program does not recognise a particular character, it must output a '?' in the appropriate position.

Caution: the output format specified above overrides the standard output requirements specified in the rules, which require separator spaces in output.

Scoring:

The score will be given as the percentage of characters correctly recognised.

Sample files:

Incomplete sample showing the beginning of FONT.DAT (space and 'a'). Sample IMAGE.DAT, showing an 'a' corrupted 'a').

FONT.DAT	IMAGE.DAT
----------	-----------

Πρόβλημα 2. Map labelling

You are a cartographer's assistant, and have been given the difficult task of writing the names of cities onto a new map.

The map is a grid of 1000 x 1000 cells. Each city occupies a single cell on the map. City names are to be placed on the map in rectangular boxes of cells. Such boxes are called labels.

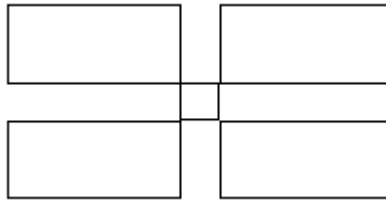


Fig. 1: A city with the four possible positions of its label

The placement of labels must satisfy the following constraints:

1. A city's label must appear in one of four positions with respect to the city as illustrated in figure 1.
2. Labels must not overlap each other.
3. Labels must not overlap cities.
4. Labels must completely fit on the map.

Each label contains all the letters in a city name plus a single space. For each city name the width and height of its letters will be given; the single space has the same size.

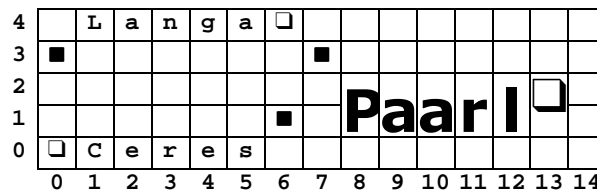


Fig. 2: Section of a map

The leftmost column of the map has horizontal index 0 and the bottom row has vertical index 0. Figure 2 shows the bottom left section of a map for the cities Langa at (0,3), Ceres at (6,1) and Paarl at (7,3). All the labels are validly placed, but this is not the only valid placement.

Task:

Your program must read the locations of the cities on the map, followed by their letter dimensions and names. The program must then place as many labels on the map as it can without violating the constraints above, and output the locations of the labels that have been placed.

Input:

The input file starts with a line containing an integer (N) giving the number of cities on the map. For each city there is an input line with

- the city's horizontal index (X) ,
- the city's vertical index (Y),
- the integer width (W) for each character in the city name,
- the integer height (H) for each character in the city name, and
- the city name itself.

City names are all single words. The number of cities is at most 1000.

No city name will be longer than 200 letters.

Sample input:

MAPS.DAT	Explanation
3	N=3
0 3 1 1 Langa	X=0, Y=3, W=1, H=1
6 1 1 1 Ceres	
7 3 1 2 Paarl	

Output:

Your program is required to output N lines. Each line must contain the horizontal index followed by the vertical index of the top left cell of the city's label. If your program is unable to place a label for a city, it must output -1 -1. These lines should be output in the same order as the cities are given in the input file. You must put a single space between the two numbers.

Sample output:

MAPS.OUT	Explanation:
1 4	Langa's label is at (1,4)
0 0	Ceres's label is at (0, 0)
8 2	Paarl's label is at (8, 2)

Scoring:

For each set of the test data:

- The score will be given as the percentage of city names placed by your program with respect to an excellent solution of the organisers.
- The minimum score is 0% and the maximum score is 100%.
- If any label violates the constraints, your program will score 0.
- If labels do not match the cities given, your program will score 0.

Problem 3. Stacking containers

The Neptune Cargo Company operates a container storage depot. Its container storage depot accepts containers for storage and subsequent removal.

Containers arrive at the depot for storage every hour on the hour. They stay at the depot for a positive integer number of hours. When a container arrives, its documentation contains the expected time when it will be removed. The first container arrives at time 1. The actual time a container is requested to be removed may ultimately be before or after the expected time by no more than 5 hours.

In this problem, the time in hours is expressed as an increasing positive integer which will not exceed 150.

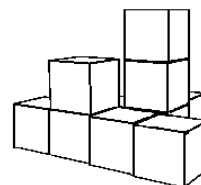
A crane (lifting and moving apparatus) operates above the storage space, moves containers in and out of the storage space, and sometimes rearranges them inside the storage space. The crane may operate in space above the defined storage space.

Task:

You are required to write a program which has a good strategy for accepting, storing and removing containers. A good strategy is one that minimizes the total number of moves that the crane makes.

The depot is a rectangular space. The length (X), width (Y) and height (Z) of the space are made available to the program.

Each container is a 1 x 1 x 1 cube. Containers are stacked on top of other containers or the floor. The crane can only move the top container of a stack.



Moving a container from one location to any other location is always one crane move. All crane moves take place between container arrivals and removals. Crane moves are instantaneous.

When the depot becomes full, your program must refuse to accept any more containers. Your program may become less efficient or unable to continue when the depot is nearly full. Your program may refuse to accept new containers at any time.

Input:

Your program is required to interact with a simulation module which will provide data, and to which your program must submit actions and messages. The depot will be empty when your program starts.

During your program testing, the library will return meaningful values for a small fixed set of test data.

Each container is identified by a unique positive integer.

Your program may call the following functions at any time:

```
int GetX();
function GetX: integer;
DECLARE FUNCTION GetX CDECL ()
    Returns length of storage space (integer).
```

```
int GetY();
```

```
function GetY: integer;
DECLARE FUNCTION GetY CDECL ()
    Returns width of storage space (integer).
```

```
int GetZ();
function GetZ: integer;
DECLARE FUNCTION GetZ CDECL ()
    Returns height of storage space (integer).
```

X,Y,Z will not exceed 32.

The following functions provide information on the action sequence (container arrivals and removals). The arrivals take place on the hour, and removal requests are received between hours. Thus, for time-keeping purposes, each arrival marks the passing of one hour.

```
int GetNextContainer();
function GetNextContainer: integer;
DECLARE FUNCTION GetNextContainer CDECL ()
    Returns a positive integer container number of the next container to be stored or retrieved. If there are no more containers to be stored or retrieved, returns 0, indicating your program should terminate, even if containers are still in the warehouse.
```

```
int GetNextAction();
function GetNextAction: integer;
DECLARE FUNCTION GetNextAction CDECL ()
    Returns an integer representing the action to take: 1 to store a new container, 2 to remove a container.
```

```
int GetNextStorageTime();
function GetNextStorageTime: integer;
DECLARE FUNCTION GetNextStorageTime CDECL ()
    Returns time in hours (since the start) when the container is expected to be removed. This value is for planning purposes for your program; the actual removal request might come at a slightly different time, which will differ by not more than 5 hours. This function only returns a meaningful value when GetNextAction returns 1.
```

The order in which the above three functions is called does not matter.

Consecutive calls to GetNextContainer, GetNextAction and GetNextStorageTime will always return information about the same container until the container is refused, stored or removed, at which point the above functions will return information about the next container.

Output:

Once your program has found out the information it needs about the next container, use the following functions to manipulate the storage depot:

```
int MoveContainer(int x1, int y1, int x2, int y2);
function MoveContainer(x1, y1, x2, y2: integer): integer;
DECLARE FUNCTION MoveContainer CDECL (BYVAL x1 AS INTEGER, BYVAL y1 AS INTEGER, BYVAL x2 AS INTEGER, BYVAL y2 AS INTEGER)
```

Move the container on the top of the stack at x1, y1 to the top of the stack at x2, y2.
Returns 1 if the action is valid, 0 if the action is illegal (i.e. impossible).

```
void RefuseContainer();
procedure RefuseContainer;
DECLARE SUB RefuseContainer CDECL ()
    Refuse to accept the incoming container.
```

```
void StoreArrivingContainer(int x, int y);
procedure StoreArrivingContainer(x, y: integer);
DECLARE SUB StoreArrivingContainer CDECL (BYVAL x AS INTEGER, BYVAL y AS INTEGER)
    Store the incoming container at the top of the stack at position x, y.
```

```
void RemoveContainer(int x, int y);
procedure RemoveContainer(x, y: integer);
DECLARE SUB RemoveContainer CDECL (BYVAL x AS INTEGER, BYVAL y AS INTEGER)
    Remove the container on the top of the stack at x, y from the depot.
```

If your program cannot carry out the required action, it should terminate.

Illegal moves are ignored by the library, and have no effect on the simulation state or scoring.

Your program is NOT required to write any output to a file. The library with which your program interacts will write a log file of actions. This file is used for evaluation.

Sequencing:

Your program should get information about the next container request. It should then move containers with the crane if desired and subsequently store, remove or refuse the action request.

Library:

A library called *StackLib* is provided which you must link to your code.

The standard C and C++ libraries contain this library and will automatically be linked to your program when you include the appropriate header file.

If you are using QuickBasic you must include the library by typing

```
QB /L STACKLIB
```

Sample source code files are present in the task directory named TESTSTK.BAS, TESTSTK.PAS, TESTSTK.CPP, and TESTSTK.C.

Scoring:

The program will be tested with several data sets and for each data set, its performance will be scored against the most efficient solution known to us, using the following indicators:

- Total number of crane moves by your program.
- A penalty of 5 moves is imposed for every container refused.
- A penalty of 5 moves is imposed for each container not stored and removed (i.e. if the program terminates normally before the entire operation is complete).
- The total score will be calculated relative to the best known solution.
- If the program makes more than twice the number of operations necessary, it scores 0.
- The minimum score is 0%, and the maximum score is 100%.

1η Βαλκανιάδα Πληροφορικής 1993

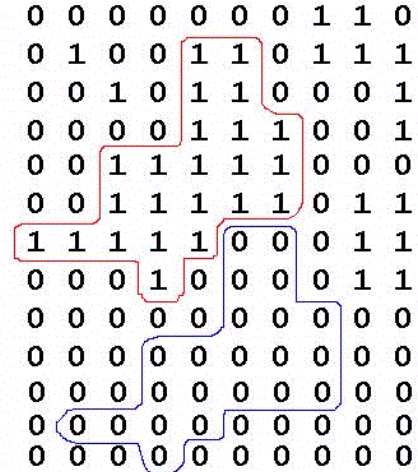
Η πρώτη Βαλκανιάδα Πληροφορικής έγινε στην Κωνσταντζα της Ρουμανίας, από τις 24 έως τις 29 Μαΐου 1993.

Ημέρα 1η

Πρόβλημα 1. Σχέδια στο χαρτί (40 πόντοι)

Ας θεωρήσουμε έναν διδιάστατο πίνακα A με m γραμμές και n στήλες, του οποίου τα στοιχεία είναι μηδέν (0) ή ένα (1), το μηδέν (0) για να παριστάνει το φόντο (χαρτί), και το ένα (1) για να παριστάνει το χρώμα ενός σχεδίου (μελάνι). Το σχέδιο γίνεται από διάφορα αντικείμενα. Με τον όρο "αντικείμενο" εννοούμε ένα σύνολο από στοιχεία του ίδιου τύπου μελάνης (1), όπου το κάθε στοιχείο έχει τουλάχιστον έναν "γείτονα" στις διευθύνσεις επάνω, κάτω, αριστερά, δεξιά ή σε κάποια διαγώνια διεύθυνση. Οποιαδήποτε δύο αντικείμενα, διαχωρίζονται από στοιχεία του τύπου "χαρτί" (0).

- Βρείτε το ορθογώνιο παραλληλόγραμμο της ελάχιστης περιοχής που περιέχει το μεγαλύτερο αντικείμενο (δηλαδή, αυτό του οποίου ο αριθμός των στοιχείων μελάνης (1) είναι μέγιστος).
- Τοποθετείστε το αντικείμενο που βρήκατε στο (α) στις θέσεις που δεν καλύπτονται από άλλα αντικείμενα, έτσι ώστε να τοποθετηθεί σε όσες περισσότερες θέσεις γίνεται, χωρίς να καταστρέψει άλλα αντικείμενα. (25 βαθμοί)



Πρόβλημα 2. Master Mind (30 πόντοι)

Δύο παίκτες - ο υπολογιστής και εσύ - παίζουν ως εξής. Σκέφτεσε έναν συνδυασμό τεσσάρων χρωμάτων (όχι αναγκαστικά διαφορετικά) από έξι πιθανά χρώματα. Ο υπολογιστής (το πρόγραμμά σας δηλαδή) πρέπει να βρει τη σειρά αυτή χρησιμοποιώντας πληροφορίες που προέρχονται από τις απαντήσεις σου. Θα απαντάς στις ερωτήσεις του υπολογιστή, μετά από κάθε νέο δημιουργημένο συνδυασμό. Οι μόνες πιθανές ερωτήσεις είναι :

Πόσα χρώματα είναι σωστά, αλλά δεν βρίσκονται στη σωστή θέση;
Πόσα χρώματα είναι σωστά και βρίσκονται στη σωστή θέση;

Παράδειγμα

Εστω ο συνδυασμός σου είναι 4655. Ενας πιθανός τρόπος να βρεί τον συνδυασμό είναι :

Υπολογιστής	Η απάντησή σου	
1234	a) 1	b) 0
5156	a) 2	b) 1
6165	a) 1	b) 1
5625	a) 1	b) 2
5653	a) 1	b) 2
4655	a) 0	b) 4

Βρείτε μια σωστή ακολουθία από συνδυασμούς έτσι ώστε να λύσετε το πρόβλημα. (15 βαθμοί).
Βρείτε, αν είναι δυνατόν, μία λύση σε έξι το πολύ προσπάθειες (15 βαθμοί).

Πρόβλημα 3. Προσθετική ακολουθία (30 πόντοι)

Μια ακολουθία από θετικούς ακεραίους a_1, a_2, \dots, a_n ονομάζεται "προσθετική αλυσίδα" μήκους n , αν για κάθε $k, 1 < k \leq n$, υπάρχουν δείκτες i και j ($1 \leq i \leq j \leq n$), τέτοιοι ώστε $a_k = a_i + a_j$. Γράψτε ένα πρόγραμμα το οποίο για $a_1=1$ και με τον τελευταίο αριθμό να διαβάζεται από το input, να βρίσκει μία προσθετική αλυσίδα με ελάχιστο μήκος.

Παράδειγμα

Η ακολουθία 1,2,4 είναι μία προσθετική ακολουθία μήκους 3.

Ημέρα 2η

Πρόβλημα 1.

Imagine you are at the social get-together with at most 500 guests. The host invites you all to have dinner. There are several tables in the dining room. The way the guests sit down at the tables is the following: each one sitting not alone at a certain table must know at least one person sitting at the same table and no one else sitting at other tables.

It is supposed that if a person knows a second person, the second one knows the first person too. No one introduces himself to the other persons sitting at the same table. That means that if two persons are sitting at the same table, but initially they do not know each other, they will not know each other afterwards either.

<Picture>(a) Determine how many tables are necessary and the persons sitting at each table. (20 points)
At each table there is only one person who will talk to the waiter; he/she is called the leader of the table. Each person relays his wishes concerning the menu to the persons he knows. The time allocated to each person to relay his wishes to each person he knows is supposed to be the same for each person.

<Picture>(b) Determine the most suitable person as leader of the table in order to receive information from all the persons sitting at the table in the shortest possible period of time; produce at output the leader of each table and the corresponding period of time. (20 points)

Afterwards, the host wishes to unify tables. For this purpose, he calls some friends. Each of them, when coming, is introduced to the leaders of two tables, links the tables, sits down at the new formed table and becomes the leader of this table.

<Picture>© What is the order of linking the tables in this way, so that at last all tables are unified into a single one and the conditions of point b) are satisfied? Specify the minimum necessary period of time for the leader to get information from all the other persons. (30 points)

After the complete linking, the friends of the host are leaving and the tables get their initial structure until the end of the dinner. When the dinner is over, the persons start leaving the tables.

<Picture>(d) Determine, for each table, the minimum number of persons and the order in which they are leaving the table, until the persons who are still at the table do not know each other. (30 points)

<Picture>Example

Suppose the number of persons is 8 and

person 1 knows persons 2 and 3;

person 2 knows persons 1 and 4;

person 3 knows persons 1 and 4;

person 4 knows persons 2 and 3;

person 5 knows person 6;

person 6 knows persons 5 and 7;

person 7 knows person 6;

person 8 does not know other persons;

A valid input is: 8

1 2

1 3

2 4

7 6

4 3

5 6

0 0

A valid output is:

a) There are 3 tables.

Table 1: 1 2 3 4

Table 2: 5 6 7

Table 3: 8

b) Leaders.

Table 1: 2

Table 2: 6

Table 3: 8

c)

6 8 New leader: 9

2 9 New leader: 10

Period of time: 3

d) Persons leaving.

Table 1: 2 3

Table 2: 6

Table 3:

2η Βαλκανιάδα Πληροφορικής 1994

Η δεύτερη Βαλκανιάδα Πληροφορικής έγινε στη Θεσσαλονίκη από 1 έως 6 Οκτωβρίου 1994.

Ημέρα 1η

Πρόβλημα 1. (25 points)

We wish to find all the numbers x (where $10 \leq x \leq 65535$) which:

- a) are prime numbers, and
- b) are Fibonacci numbers of order two (see the hint), and
- c) at least one different prime number is produced by permuting the digits of x .

The output should consist of a number of lines equal to the number of elements in the desired list. Each line should contain the number x and the prime number produced from x by permuting the digits of x , separated by a space.

<Picture>Example

The output should start as follows:

13 31

.
.
.

<Picture>Hint A prime number x may be divided only by 1 and x (1 is not a prime).

The sequence of order two Fibonacci numbers is defined as follows:

$F_0=0$

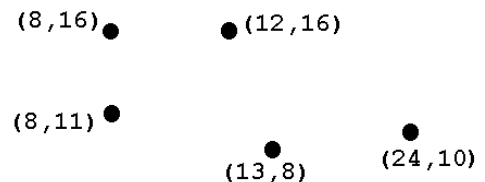
$F_1=1$

$F_i=F_{i-1}+F_{i-2}$, for all $i > 1$

Πρόβλημα 2. (40 πόντοι)

Computer networking requires that the computers in the network be linked. This problem considers a linear network in which the computers are chained together so that each one is connected to exactly two others except for the two computers on the ends of the chain which are connected to only one computer. In the picture shown below, the computers are the black dots and their locations in the network are identified by planar coordinates (relative to a coordinate system not shown in the picture).

For various reasons it is desirable to minimize the length of the cable used. The problem is to determine how the computers should be connected in such a chain to minimize the total length of cable needed. In the installation being constructed, the cable will run beneath the floor, so that the amount of cable used to join 2 successive computers will be equal to the distance between the computers plus 10 additional meters of cable to connect from the floor to the computers and provide some slack for ease of installation.



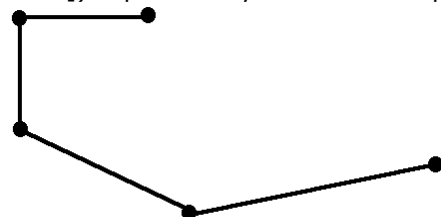
Example

Input

The input file will represent a data set beginning with a line consisting of a single number indicating the number of computers in the network (at least 2 and at most 10 computers). Then, in successive lines the coordinates are given as pairs of integer numbers (in the range $[0..100]$) separated by one or more space, e.g.

```
5
8 11
8 16
12 16
13 8
24 10
```

The picture shows the optimal way of connecting the computers shown above.



Output

The output should consist of n lines, where n is the number of computers. The first line should give the total length of cable needed. Subsequent lines should comprise of three numbers, l , s , e (from: length, start, end). The first number, l , should depict the length of the cable to be cut to connect the two

computers, s and e. The values of s and e should represent the order of computers in the input. Any direction for traversing the network from the one end to the other is valid. For example,

```
66.01
14      3      2
15      2      1
15.83   1      4
21.58   4      5
```

Πρόβλημα 3. (35 πόντοι)

Recycling glass requires that the glass be separated by color into one of three categories: brown, green, and clear glass. You are given three recycling bins, each containing a specified number of brown, green and clear bottles. In order to be recycled, the bottles will need to be moved so that each bin contains unicolored bottles. The problem is to minimize the number of bottle movements assuming that the bin capacity is infinite.

The first line of the input is a single integer denoting the number of successive lines. Each successive line consists of 9 integers separated by space. The first three integers on a line represent the number of brown, green and clear bottles respectively in the bin #1, the second three integers represent the number of brown, green and clear bottles respectively in the bin #2, whereas the third three integers represent the number of brown, green and clear bottles respectively in the bin #3.

For each line of input there will be one line of output containing:

- a) a string consisting of three characters: B, G, and C (standing for the three colors), representing the content of three numbered bins, and
- b) an integer denoting the minimum number of bottle movements. If more than one order of B, G, and C yields the minimum number of movements, then you should give the first string in alphabetical order.

Παράδειγμα

Είσοδος

```
2
1 2 3 4 5 6 7 8 9
5 10 5 20 10 5 10 20 10
```

Εξοδος

```
BCG 30
CBG 50
```

Ημέρα 2η

Πρόβλημα 1.

Assume an $n \times n$ table (where $n \leq 10$) containing positive integer numbers $1 < x \leq 40$.

a) (30 points)

Replace each number x with a number y , where y is the maximum number of prime numbers which sum to x , so that any prime number appears once in the sum with at most one and only one exception of a prime number which may appear twice. Have, also, in mind that 1 is not prime. For example:

$2=2$	$f(2)=1$
$3=3$	$f(3)=1$
$4=2+2$	$f(4)=2$
$5=3+2$	$f(5)=2$
$6=3+3$	$f(6)=2$
$7=3+2+2$	$f(7)=3$

Give the resulting table in the output.

b) (10 points)

After each element x has been replaced with y , determine the maximal number of y elements which are connected horizontally, vertically or diagonally forming a concrete area, and give this y value. Also, determine, the smallest enclosing rectangle which can be defined by giving the coordinates (i.e. row and column) of the upper leftmost and lower rightmost corners of the enclosing rectangle.

c) (25 points)

In general, the area found in (b) is not a square or even a rectangle. Determine the minimum number of non-overlapping squares of any size (e.g. 1×1 , 2×2 , 3×3 , etc.) it consists of.

d) (20 points)

Try to maximize the area determined in (b) with the minimum number of element swappings of the table produce in question (a). Give, also, the coordinates of the elements which are swapped. Show in the output the new form of the table obtained.

e) (15 points)

Determine the y element that appears most often in the table produced in question (a). In case of a tie (e.g. two or more y elements with the same value) choose the y element with the smallest value. Try to collect all its instances nearby (i.e. so that the are connected horizontally, vertically or diagonally) and give the minimum number of swappings performed. Give, also, the coordinates of the elements which are swapped. Illustrate the new form of the table obtained. Beware that the swappings should be applied on the table produced in question (a).

Example

Input

The first line consists of a single integer number denoting n, the length of the table side. Then, is successive lines the rows are given containing the integer numbers separated by one space, e.g.

```
6
19 19 12 12 12 19
19 12 7 12 12 7
2 4 7 7 12 19
19 12 12 12 12 19
19 19 19 19 5 19
30 6 30 5 12 19
```

Output

a) After replacing x with y, the output should be as follows:

```
5 5 4 4 4 5
5 4 3 4 4 3
1 2 3 3 4 5
5 4 4 4 4 5
5 5 5 5 2 5
6 2 6 2 4 5
```

b) The output should be given in three separate lines as follows:

```
11
4
(1,2),(4,5)
```

where the first line gives the maximal number of y connected elements, the second line contains the y value, and the third line contains the coordinates of the enclosing rectangle.

Hint: you may avoid the parentheses and give the coordinates separated by commas. The same hint holds for the output of question (d) and (e).

c) The output consists of a single number:

```
8
```

which stands for the minimum number of non-overlapping square in the area of (b).

d) The output should be as follows:

```
1
(5,5),(6,5)
5 5 4 4 4 5
5 4 3 4 4 3
1 2 3 3 4 5
5 4 4 4 4 5
5 5 5 5 4 5
6 2 6 2 2 5
```

where the first line gives the minimum number of swappings, and the second line gives the coordinates of the swapped element(s). The next n lines contain the new form of the table.

e) The output should be as follows:

```
5
2
(1,2),(3,1)
(1,6),(5,5)
5 1 4 4 4 2
5 4 3 4 4 3
5 2 3 3 4 5
5 4 4 4 4 5
5 5 5 5 5 5
6 2 6 2 4 5
```

where the first line stands for the most often appearing element, the second line for the minimum number of swappings, the next lines give the coordinates of the two swappings. Finally, the last n lines illustrate the new form of the table

Attention: Your program will begin with a question:

«Do you wish to solve part (a) ?»

If the answer is YES the the program will start assuming an input file with the name INPUT1.TXT. Otherwise, part (b) will start assuming an input file under the name INPUT2.TXT.

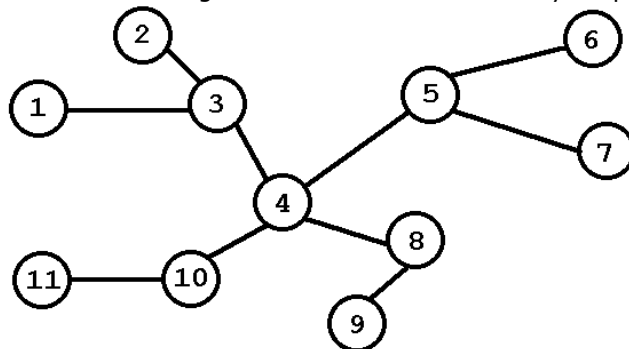
3η Βαλκανιάδα Πληροφορικής 1995

Η τρίτη Βαλκανιάδα Πληροφορικής έγινε στη Βάρνα της Βουλγαρίας, από 5 έως 11 Οκτωβρίου 1995.

Day 1 - Problem 1 (Reservoirs)

A system for fuel oil supply consists of n ($n \leq 200$) numbered reservoirs with given volumes (≤ 100) and $n-1$ tubes, assumed with zero volumes, that join pairs of reservoirs. The system is connected and it is filled up through one of the reservoirs s called source in the following manner. The sources s stay empty and pumps fuel oil so that per unit of time a unit of fuel goes through each of the tubes t_1, t_2, \dots, t_k joined to s , until all reservoirs except the source are filled up. Therefore the process of filling takes time equal the maximum of V_1, V_2, \dots, V_k , where V_i , for $i=1,2,\dots,k$ is the sum of the volumes of the reservoirs that are filled up through tube t_i .

Given the system of the described type, write a program that finds one optimal reservoir (source) from which the process of filling takes minimum time.



Input

The program input will be given in an ASCII file name INPUT.1 with the following format:

```
line 1 :          n      - an integer denoting the number of the reservoirs;
line i for i=2,...,n+1 : vi_1 - an integer denoting the volume of the
reservoir i-1;
line i for i=n+1,...,2n : r q - pair of integers denoting that there is a tube
between reservoir r and reservoir q. The numbers r q are separated by one space.
```

Output

The output should be one standard output (screen) and should contain two lines as follows:

The number of the optimal source is: ...

The optimal time is: ...

Example

The input data for the example are:

```
11
1
1
1
1
1
1
1
1
1
1
11
1 3
2 3
3 4
4 5
5 6
5 7
4 8
8 9
4 10
10 11
```

Και η έξοδος πρέπει να είναι:

The number of the optimal source is: 4

The optimal time is: 2

Πρόβλημα 2. Μεγάλος Αριθμός

Ο θετικός ακέραιος αριθμός n παριστάνεται από k ($40 \leq k \leq 80$) ψηφία στο δεκαδικό σύστημα, και είναι ένας μεγάλος αριθμός. Το πρόβλημα είναι να γράψετε ένα πρόγραμμα το οποίο, για έναν δεδομένο αριθμό n , να βρείσκει τον μεγαλύτερο ακέραιο τέτοιο ώστε $p^3 + p^2 + 3p \leq n$.

Είσοδος

Η είσοδος δίνεται σε ένα αρχείο ASCII που ονομάζεται INPUT.2. Η πρώτη γραμμή περιέχει τον αριθμό n με τα k ψηφία του.

Εξοδος

Η έξοδος θα δοθεί στο standard output (οθόνη) και θα περιέχει τον ακέραιο p .

Παράδειγμα

Input

```
1000000000000001000000000000003000000000000001
```

Output

```
1000000000000000
```

Ημέρα 2η

Πρόβλημα 1. Διανομές

A firm occupies a 10-floors building. After finishing the work day, clerk John has to carry identical packages between floors. He starts and ends this job on floor 1 and can carry only one package.

Write a program that finds the minimum number m of the climbed floors and with the corresponding way of traveling.

Input

The required deliveries are in the text file INPUT.TXT as follows:

```
first line:    number of deliveries  $k \leq 50$ ;  
next lines:   2 numbers that determine a delivery, i.e. the floor sending  
              and the floor receiving a package, for example:  
              7  
              1 10  
              3 2  
              4 3  
              6 7  
              8 9  
              3 4  
              4 3
```

Output

The deliveries are numbered in order they appear in the file: 1,2,...,k.

Output is in the file in the following format:

```
first line:    number  $m$ ;  
next lines:   description of John's movements, which are described by  
              triples:  $b_i, e_i, d_i$ , where,  
               $b_i$  - number of sending floor;  
               $e_i$  - number of receiving floor;  
               $d_i$  - number of delivery, which package John has to carry  
from floor  $b_i$  to floor  $e_i$ . Number 0 (zero) means  
that John carries no package at this step.
```

Here is one possible answer for the given example. The output OUTPUT.TXT consists of:

```
12  
1,6,1  
6,7,4  
7,6,0  
6,10,1  
10,8,0  
8,9,5  
9,4,0  
4,3,4  
3,4,6  
4,3,7  
3,2,2  
2,1,0
```

Πρόβλημα 2. Πρωτάθλημα Baseball

One of the tasks of the Ministry of Sports of a Balkan country is to organize a baseball championship of n teams (numbered from 1 to n). The championship must consist of no more than n rounds and each two teams must play against each other exactly once. Write a program that produces a schedule for the championship.

Input

The number of teams n ($1 \leq n \leq 24$) will be entered from the console.

Output

The output is a table of integers with n rows. The j -th integer in the i -th row is the number of the team which plays against team i in the j -th round. If team i is idle in the j -th round then this number is 0.

Sample input

```
3      4
```

Sample output

```
2 3 0      2 3 4
1 0 3      1 4 3
0 1 2      4 1 2
           3 2 1
```

Πρόβλημα 3. Περιστρεφόμενοι Κύλινδροι

A connected transmission is constructed by n ($n \leq 30$) cylindrical units, labeled by $1, 2, \dots, n$ in such a way that some pairs of units come to contact. If the units i and j contact and one of them is made going round in one of the two possible directions (Clockwise or Opposite), then the other begins to go round in the opposite direction. If a unit going round in one direction then the transmission blocks up.



You have to write a program which for a given transmission determines its behavior when a given unit i is made to go round in a given direction: blocks up or works normally. If the transmission works normally the program prints a list of units going round with the corresponding directions. Else the program has to determine the minimum number of units different from i to be removed from the blocked transmission so that all remaining units are going round when the unit i is made to go round.

Είσοδος

Input will be in ASCII file with the following format. The first line contains the number n of units and the number m of contacting pairs of units. Each of the next m lines contains two labels of pairs of contacting units. In the last line is given the number i of a unit and one of the characters C or O defining the direction in which that unit has to be made to go round. For the transmission of the two figures the input will be:

```
5 5      3 3
1 2      1 2
1 3      1 3
2 4      2 3
3 4      1 O
3 5
1 C
```

Εξοδος

The output will be on the standard output (screen) and consist of 3 or 4 lines. The first line describes the behavior of the system in one letter: W (for «Works») or B (for «Blocks»). If the system works then the second line is a list of units rotating Clockwise, whereas the third line is a list of units rotating Opposite direction after the removing is done and an additional line of output is a list of Removed units. For the above example the output should be:

```
W      B
C 1 4 5    C 2
O 2 3      O 1
           R 3
```

4η Βαλκανιάδα Πληροφορικής 1996

Η τέταρτη Βαλκανιάδα Πληροφορικής έγινε στην Λευκωσία της Κύπρου, από 19 έως 25 Οκτωβρίου 1997.

Ημέρα 1η

Πρόβλημα 1. Το ενυδρείο

Αν θέλουμε να φτιάξουμε ένα ενυδρείο από εξωτικά ψάρια, πρέπει να συμβουλευτούμε κάποιον ειδικό, για να αποφύγουμε τις περιπτώσεις που δύο είδη ψαριών δεν μπορούν να ζήσουν μαζί. Ο λόγος είναι η "ασυμβατότητα" μεταξύ των ειδών των ψαριών.

Δίνεται ο αριθμός των ειδών των ψαριών και ένα διαθέσιμο χρηματικό ποσό για να ξοδέψουμε. Βρείτε τον μέγιστο αριθμό από διαφορετικά είδη ψαριών που μπορούμε να βάλουμε στο ενυδρείο. Για τον αριθμό αυτό βρείτε τα ποσό που μπορούμε να ξοδέψουμε, δεδομένου ότι αγοράζουμε μόνο ένα ψάρι από κάθε είδος.

Είσοδος (αρχείο INPUT.TXT)

Το πρόγραμμα διαβάζει γραμμές από το αρχείο INPUT.TXT ως εξής:

- Στην πρώτη γραμμή υπάρχουν δύο ακέραιοι, χωρισμένοι με κενό. Ο πρώτος είναι το διαθέσιμο ποσό χρημάτων M ($M \leq 1000$) και ο δεύτερος είναι ο αριθμός των διαφορετικών ειδών από ψάρια F ($F \leq 30$).
- Στις επόμενες F γραμμές, υπάρχει ο αριθμός σειράς κάθε είδους ψαριού και το αντίστοιχο κόστος του είδους, χωρισμένα με ένα κενό.
- Στις επόμενες γραμμές και μέχρι το τέλος του αρχείου, το αρχείο περιέχει ζεύγη από είδη ψαριών, που δεν μπορούν να συμβιώσουν στο ενυδρείο. Το τέλος του αρχείου ορίζεται σαν μία γραμμή που περιέχει δύο μηδενικά χωρισμένα από ένα κενό (0 0).

Εξοδος (αρχείο OUTPUT.TXT)

Η έξοδος γράφεται στο OUTPUT.TXT ως εξής:

- Στην πρώτη γραμμή, δύο ακέραιοι χωρισμένοι με ένα κενό, οι οποίοι δείχνουν το μέγιστο αριθμό από ψάρια που μπορούν να μπουν στο ενυδρείο και το μέγιστο ποσό χρημάτων που απαιτείται για να αγοραστούν, αυτά τα είδη ψαριών.
- Στις επόμενες γραμμές, γράψτε τους αριθμούς σειράς των ειδών των ψαριών που θα βάλουμε στο ενυδρείο.

Παράδειγμα

Input

```
170 7
1 70
2 50
3 30
4 40
5 40
6 30
7 20
1 4
1 7
3 4
3 5
5 7
6 7
0 0
```

Output

```
4 160
2
4
5
6
```

Παρατήρηση

Στην περίπτωση που υπάρχουν πολλές λύσεις, μόνο μία πρέπει να δοθεί. Το χρονικό όριο για κάθε έλεγχο είναι 5 δευτερόλεπτα. Αριστα : 30 πόντοι.

Πρόβλημα 2. Ο λαβύρινθος 1-2-3-4

Εχουμε έναν ορθογώνιο παραλληλόγραμμο λαβύρινθο με διαστάσεις $n \times m$ ($n \leq 20$, $m \leq 20$). Η επάνω αριστερή γωνία και η κάτω δεξιά είναι μαρκαρισμένες με ένα 0, ενώ όλες οι άλλες θέσεις έχουν μέσα 1, 2, 3 ή 4. Ο σκοπός του παιχνιδιού είναι να "μπούμε" στον λαβύρινθο από την πάνω αριστερή γωνία και να βγούμε από την κάτω δεξιά, ακολουθώντας τον συντομότερο δρόμο και κινούμενοι μόνο προς τα δεξιά,

αριστερά, πάνω και κάτω. Ο κανόνας που πρέπει να ακολουθούμε είναι ότι πρέπει να μετακινούμαστε από έναν άσο (1), σε 2, σε 3, σε 4 και πάλι σε 1, κ.λ.π. με έναν "κυκλικό" τρόπο. Από την αρχική θέση μπορούμε να μετακινηθούμε μόνο σε ένα κουτάκι με άσο. Στην έξοδο μπορούμε να βρεθούμε είτε από τη γειτονική αριστερή θέση, ή από τη γειτονική επάνω θέση.

Είσοδος (αρχείο INPUT.TXT)

Η πρώτη γραμμή περιέχει τους ακεραίους n και m . Οι επόμενες n γραμμές περιέχουν m ακεραίους που παριστάνουν το λαβύρινθο.

Εξοδος (αρχείο OUTPUT.TXT)

Το αρχείο θα περιέχει δύο γραμμές, η πρώτη με τον αριθμό των βημάτων και η δεύτερη με ένα string από χαρακτήρες. Το string περιλαμβάνει τους χαρακτήρες D (για κάτω - down), U (για επάνω - up), L (για αριστερά - left) και R (για δεξιά - right), που αντιστοιχούν στα βήματα που κάνουμε στη συντομότερη διαδρομή προς την έξοδο του λαβυρίνθου.

Παράδειγμα

Input

```
5 4
0 1 2 3
3 2 1 4
4 1 2 1
1 4 3 2
2 3 4 0
```

Output

```
7
RRRDDDD
```

Παρατήρηση

Μπορείτε να υποθέσετε ότι πάντα υπάρχει μία λύση στο πρόβλημα.
Το χρονικό περιθώριο για κάθε έλεγχο είναι 5 δευτερόλεπτα.
Αρίστα : 30 πόντοι.

Πρόβλημα 3. Κορώνα - Γράμματα

Δίνεται ένας $N \times N$ πίνακας (όπου $N \leq 10$) από κέρματα. Κάποια από αυτά είναι γυρισμένα "κορώνα", κάποια "γράμματα". Ένας παίκτης θέλει να γυρίσει τα κέρματα έτσι ώστε να τα φέρει όλα γυρισμένα στα "γράμματα". Όμως, κάθε φορά που ο παίκτης γυρνάει ένα κέρμα, τότε κάθε κάθετο κέρμα-γείτονας (δηλαδή, ο πάνω, ο κάτω, ο αριστερός και ο δεξιός, αν υπάρχουν) γυρνάει και αυτό.

Το πρόβλημα είναι, αφού σας δοθεί μία αρχική κατάσταση, να καταφέρετε να θέσετε όλον τον πίνακα σε "γράμματα" κάνοντας τις λιγότερες κινήσεις. Σαν κίνηση θεωρούμε, ότι "δείχνουμε" ένα κέρμα και το αλλάζουμε, και αυτό και όλους τους "γείτονες" του.

Είσοδος (αρχείο INPUT.TXT)

Η πρώτη γραμμή περιέχει τον ακέραιο N . Κάθε μία από τις επόμενες N γραμμές περιέχουν N χαρακτήρες, H ή T που αντιστοιχούν σε κορώνα(heads) / γράμματα(tails) αντίστοιχα.

Εξοδος (αρχείο OUTPUT.TXT)

Η πρώτη γραμμή πρέπει να περιέχει τον μικρότερο αριθμό από γυρίσματα. Οι επόμενες N γραμμές θα συνθέτουν έναν πίνακα από N ακεραίους σε κάθε γραμμή. Αυτοί οι ακέραιοι θα έχουν τιμή 1 ή 0. Η τιμή 1 σημαίνει ότι το αντίστοιχο κέρμα "γύρισε" τουλάχιστον μία φορά. Η τιμή 0 σημαίνει ότι το αντίστοιχο κέρμα δεν γύρισε καθόλου.

Παράδειγμα

Input

```
3
H T T
H T T
T T T
```

Output

```
5
1 0 0
0 1 0
1 1 1
```

Παρατήρηση

Μπορείτε να υποθέσετε ότι πάντα υπάρχει μία λύση στο πρόβλημα.
Το χρονικό περιθώριο για κάθε έλεγχο είναι 20 δευτερόλεπτα.
Αρίστα : 40 πόντοι.

Ημέρα 2η

Πρόβλημα 1. Περιμένοντας για εισητήριο

The U2 rock group will give a concert in Nicosia during November 96. A queue of $n \leq 200$ U2-fans waiting to buy a ticket from a single cashier has been formed. Each person wants to buy only one ticket, whereas the cashier can sell at most two tickets to a person.

Suppose that the cashier spends t_i time units to serve the i -th fan (where $1 \leq i \leq n$). However, two consecutive fans in the queue (e.g. the j -th and the $(j+1)$ -th one ($1 \leq j \leq n-1$)) may agree so that one of them remains in the queue and the other leaves the queue, if the time r_j required by the cashier to serve the j -th and the $(j+1)$ -th fan is smaller than $t_j + t_{j+1}$. Thus, in order to minimize the cashier's time and prevent the U2-fans from being squeezed, each person in the queue tries to negotiate with his predecessor and his successor, and finally speed up the process.

Given positive integer values for n , t_i and r_j , minimize the total cashier's time. This is achieved by making pairs of consecutive persons in an optimal way. Note, however, that it is not necessary for a specific fan to be paired with a predecessor or a successor.

Input

Your program should read the input file named INPUT.TXT. Input data consists of three lines:

- The first line contains the integer n ,
- The second line contains n integers, the t_i values separated by spaces, and
- The third line contains $n-1$ values standing for the r_j values separated by spaces.

Output

The first line of the file named OUTPUT.TXT contains an integer which represents the total cashier's time. Each one of the next lines contains either 1 number or 2 numbers separated by the character +. More specifically, each line contains the number i , if the i -th fan is served alone, or $i+(i+1)$ if these two fans are served as a pair.

Example

Input

```
7
5 4 3 2 1 4 4
7 3 4 2 2 4
```

Output

```
14
1
2+3
4+5
6+7
```

Time Limit: 15 sec

Maximum Score : 35

Πρόβλημα 2. (Satellite configuration)

We want to launch a set of satellites in the sky in orbit around earth. Two satellites may transmit data to each other in a direct way, or in indirect way (i.e. via one or more intermediate satellites), or finally may have no connection at all. We say that two satellites are neighbors if they can send data to each other in a direct way.

Given a number n of satellites and a set of n integers, find out:

if there exists a configuration of n satellites such that the given set of n integers correspond to the neighbors of the satellites, •if the answer in part (a) is positive, then describe such a setting, •given the setting of part (b), conclude if we can send data from any satellite to any other one.

<Picture>Input

The input file INPUT.TXT consists of a line containing a sequence of integers separated by spaces. The first integer, a number $n \leq 20$, gives the number of satellites in orbit. The following n integers represent the required number of neighbors for each one of the satellites. Note that the order of these n numbers (which represent the number of neighbors for each satellite) is not important.

<Picture>Output

The first line of the output file OUTPUT.TXT contains the answer to part (a): a YES or a NO.

In case of a NO in part (a), then no more output is required.

In case of a YES in part (a), then a number of n lines follow containing n integers 1 or 0 separated by spaces, forming an $n \times n$ matrix. If the (i,j) -th element of this matrix is 1, then the satellites i and j are neighbors, whereas if this element is 0 then these satellites are not neighbors. The $(n+2)$ -th line of the output file corresponds to part (c) and is a YES or a NO.

<Picture>Example 1
Input 6 4 3 1 4 2 0 •Output NO

<Picture>Example 2
Input
7 4 3 1 5 4 2 1 •Output
YES
0 1 1 1 1 1 0
1 0 1 0 0 0 0
1 1 0 1 0 1 0
1 0 1 0 0 1 0
1 0 0 0 0 0 0
1 0 1 1 0 0 1
0 0 0 0 1 0
YES

Example 3
Input
6 2 3 1 1 2 1 •Output
YES
0 1 0 1 0 0
1 0 1 1 0 0
0 1 0 0 0 0
1 1 0 0 0 0
0 0 0 0 1
0 0 0 1 0
NO

Time Limit : 20
Maximum Score : 35

Πρόβλημα 3. Παράθυρα σε ένα γραφικό περιβάλλον εργασίας

A common operation in graphical environments is the handling of overlapping windows. Only the exposed portions of a window are displayed, with special routines written to determine what portions of a window are exposed, and should be dealt with. Your task is to take a list of windows, do the appropriate manipulations, and, upon request, indicate the exposed area of a given window.

Input

The program reads the input file INPUT.TXT as a series of one instruction per line in the following format:

```
w(I,X,Y,x,y)
t(I)
b(I)
e(I)
s(I)
```

In the above input 'w', 't', 'b', 'e', 's' are the commands manipulating the windows. The rest of the information represent the following data:

I is a windows id. The windows identifiers will be single characters in the range: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-+ As the identifiers are unique, it will not be possible to have two windows with the same id, open at the same moment. There will be no more than 64 windows opened at any one time. •X and Y are the upper left-hand coordinates of the window. •x and y are the lower right-hand coordinates of the window. It is assumed that in order to perform an operation on a given window, this specific window must have been created before with the 'w' command.

Explanation of window-commands :

'w': creates a window with the appropriate id, size and location. A newly created window is created on top of all existing windows.

't': brings the indicated window on the top of all other windows.

'e': erases the indicated window from the screen.

'b': puts the indicated window below all other windows.

's': shows the percentage of indicated window's exposed area, printed as a real number followed by a new line.

Output

The output file of the program OUTPUT.TXT will simply be responses to the 's' commands. For each 's' command found and processed in the input file, one output line is generated that shows in a percentage the exposed area of the indicated window.

Example

Input

```
w(a,10,132,20,12)
w(c,12,120,22,16)
w(b,8,16,124,15)
t(a)
w(d,18,93,102,20)
b(b)
b(a)
s(a)
s(b)
s@
s(d)
e(d)
e@
s(a)
s(b)
```

Output

```
29.83%
100.00%
71.92%
100.00%
99.17%
100.00%
```

Limits

The screen that all windows will be displayed on will not exceed 250x250. The thickness of windows boundaries is negligible (i.e. w(a,0,4,4,0) is a window of area 16).

Note :

Correct output values will be considered those with ñ0.02% deviation from the indicate value in the output (i.e. for 29.83% the percentages 29.81%, 29.82%, 29.84% and 29.85% will be consider correct).

Time limit per test : 5 seconds

Maximum Score : 30

Central European Olympiads in Informatics

1st CEOI 1994

Ημέρα 2η

Πρόβλημα 1. Μαύρο - άσπρο

Γράψτε ένα πρόγραμμα το οποίο διαβάζει τρεις θετικούς ακεραίους n , p , q . Αποφασίστε αν υπάρχει κάποια ακολουθία από n ακεραίους της οποίας το άθροισμα από οποιουσδήποτε p συνεχόμενους αριθμούς και το άθροισμα από οποιουσδήποτε q συνεχόμενους αριθμούς να είναι αρνητικός αριθμός. Αν η απάντηση είναι ΝΑΙ δημιουργήστε μία τέτοια ακολουθία.

Οι τιμές n , p , q διαβάζονται από το πληκτρολόγιο. Η έξοδος του προγράμματος αποτελείται από n αριθμούς που γράφονται στην οθόνη.

ΠΑΡΑΔΕΙΓΜΑΤΑ

για τα δεδομένα:

$n=4$
 $p=2$
 $q=3$

το αποτέλεσμα είναι:

NO

για τα δεδομένα:

$n=6$
 $p=5$
 $q=3$

το αποτέλεσμα είναι:

YES
-3 5 -3 -3 5 -3

Πρόβλημα 2. Μονό – ζυγό δίκτυο

Ας θεωρήσουμε μία χώρα με N πόλεις. Δίνεται ένα σύστημα από δρόμους που υπάρχουν μεταξύ των πόλεων. Ο δρόμος που ενώνει απ' ευθείας δύο πόλεις θεωρείται ότι έχει μήκος 1. Το σύστημα αυτό ονομάζεται μονό-ζυγό, αν υπάρχουν δύο πόλεις που συνδέονται με έναν δρόμο περιπτού (μονού) μήκους και με έναν δρόμο άρτιου (ζυγού) μήκους.

α) βρείτε αν το σύστημα είναι μονό-ζυγό

β) αν η απάντηση στο α) είναι αρνητική βρείτε ένα υποσύνολο X των πόλεων που έχει το μέγιστο αριθμό πόλεων και ικανοποιεί την ακόλουθη συνθήκη: για κάθε δύο πόλεις στο X , αν υπάρχει διαδρομή που τις ενώνει τότε το μήκος της είναι μονό.

ΕΙΣΟΔΟΣ

Στο INPUT.TXT υπάρχει ο αριθμός των πόλεων N , και στις επόμενες N γραμμές δύο αριθμοί I, J που σημαίνουν ότι η πόλη I και η πόλη J συνδέονται απ' ευθείας. Το N είναι το πολύ 300. Η τελευταία γραμμή των ζευγαριών θα είναι δύο μηδενικά.

ΕΞΟΔΟΣ

Τυπώστε τα αποτελέσματα στην οθόνη σε μία προφανή εικόνα.

ΠΑΡΑΔΕΙΓΜΑ

αν η είσοδος είναι:

5
1 2
2 3
3 4
4 5
5 1
0 0

η έξοδος θα είναι:

YES

αν η είσοδος είναι:

3

```
1 2
0 0
τότε η έξοδος θα είναι:
NO
X has 2 elements
X: 2 3
```

Πρόβλημα 3. Αριθμητικές Παραστάσεις

Μία αριθμητική παράσταση περιέχει προσθέσεις και πολλαπλασιασμούς. Ο χρόνος που χρειάζεται για να υπολογίσουμε μία πρόσθεση (+) είναι p και ο χρόνος που χρειάζεται για να υπολογίσουμε έναν πολλαπλασιασμό (*) είναι q . Ο χρόνος που χρειάζεται για να υπολογίσουμε την παράσταση $A \circ B$ είναι ίσος με τον χρόνο που χρειάζεται για να υπολογίσουμε την πράξη \circ , συν τον μέγιστο χρόνο από τους χρόνους που χρειάζονται για να υπολογίσουμε τις υποπαραστάσεις A και B .

Στις παραστάσεις παίρνουν μέρος μεταβλητές που αποτελούνται από έναν χαρακτήρα και ο χρόνος υπολογισμού των είναι ίσος με 0.

Γράψτε ένα πρόγραμμα το οποίο διαβάζει δεδομένα από ένα αρχείο που περιέχει τους αριθμούς p και q και παραστάσεις, κάθε παράσταση σε ξεχωριστή γραμμή. Οι παρενθέσεις χρησιμοποιούνται για να ορίσουν τη σειρά των πράξεων. Για κάθε παράσταση ζητείται:

- α) να βρείτε και να τυπώσετε το χρόνο που χρειάζεται για να υπολογιστεί.
- β) να βρείτε και να τυπώσετε μία ίση αριθμητική παράσταση με το μικρότερο δυνατό χρόνο υπολογισμού, και το χρόνο αυτό στην επόμενη γραμμή.

Οι ισοδύναμες παραστάσεις που είναι δυνατόν να γίνουν είναι:

```
x+y=y+x, x*y=y*x (αντιμετάθεση)
x+(y+z)=(x+y)+z, x*(y*z)=(x*y)*z (προσεταιριστικότητα)
```

Τα αποτελέσματα θα τυπωθούν στο αρχείο OUTPUT.TXT με μία κενή γραμμή να ξεχωρίζει τα αποτελέσματα δύο παραστάσεων.

ΠΑΡΑΔΕΙΓΜΑ

αν το INPUT.TXT περιέχει:

```
1 1
((a+(b+(c+d)))*e)*f
(((a*b)*c)*d)+e)+(f*g)
```

τότε το OUTPUT.TXT θα πρέπει να περιέχει:

```
5
((a+b)+(c+d))*e*f
3

5
(((a*b)*(c*d))+e)+(f*g)
4
```


Ημέρα 1η

Problem 1. Idle machine

A computer records the starting and finishing time of each application program in the measure of 1/100 seconds. A daily statistic is a set of pairs (a, b) of non negative integers such that $0 < a \leq b < 8640000$. The pair (a, b) means that a program started at time a and finished at time b. The time points a and b belong to busy time. Arbitrary number of programs can run in parallel.

Write a program that computes

- a) the length (b-a+1) of the largest closed time interval [a, b] when the machine was idle and
- b) the length (d-c+1) of the largest closed time interval [c, d] when the machine executed at least one program. A time point t belongs to the idle time if there is no program running at time t.

Input data

The first line of text file DAY1A.DAT contains N, the number of the intervals ($1 \leq N \leq 1000$). Each of the rest N lines of the file contains two non negative integers, the starting and finishing time of a program.

Output Data

The text file DAY1A.SOL has to contain the following two integers in this order:

- a) The length (b-a+1) of the largest interval of time [a, b] when the machine was idle.
- b) The length (d-c+1) of the largest closed time interval [c, d] when the machine executed at least one program.

Maximum number of points: 25.

Example input:

```
9
30000 35000
10000 20000
15000 16000
40000 44000
77000 220000
13000 41000
60000 67000
50000 55000
65000 70000
```

Example output:

In our example input the DAY1A.SOL file contains the following two lines:

```
8419999
143001
```

Problem 2. Alarm chain

The students of a class have decided to form an alarm chain. Every student chooses a unique successor, to whom he directly delivers received messages. Whenever a student receives a message forwards it to his or her successor.

Such an assignment is called an alarm chain if the following holds. Suppose somebody sends a message to his or her successor who in turns transmits the message to his or her successor, etc. The message eventually arrives to every student, including the starting student.

Clearly, not every assignment is alarm chain. Write a program which for an arbitrary input assignment, computes the number of necessary modifications that transforms the input assignment to an alarm chain.

Input data

The first line of the text file DAY1B.DAT contains N, the number of the students ($1 \leq N \leq 256$). Each of the following N lines contains two names (strings) separated by the character '>'. The second name terminated by end-of-line character. Names are at most 20 characters long. A line of the form A>B means that the successor of student A is B, i.e. A directly delivers messages to B.

Output Data

Write the minimal number of necessary modifications to the text file DAY1B.SOL as an integer.

Maximum number of points: 35.

Example input:

```
10
Anita>Peter
Andrew>Julia
David>Andrew
Natalie>Gabriella
Edith>David
Peter>Anita
Gabriella>Julius
Adam>David
Julia>Gabriella
Julius>Julia
```

Example output:

```
4
```

Problem 3. Fair sharing

Two brothers, Alan and Bob want to share a set of gifts. Each of the gifts should be given to either Alan or Bob, and none of the gifts can be split. Each gift has a positive integer value. Let A and B denote the total value of gifts received by Alan and Bob, respectively. The aim is to minimise the absolute value of the difference A-B. Write a program which computes the values A and B.

Input data

The first line of the text file DAY1C.DAT contains N, the number of the gifts ($1 \leq N \leq 100$). The rest of the input contains N positive integers, the values of the gifts. Each value is ≤ 200 .

Output Data

Write the two integers A and B to the text file DAY1C.SOL.

Maximum number of points: 40.

Example input:

```
7
28 7 11 8 9 7 27
```

Example output:

```
48 49
```

Ημέρα 2η

Πρόβλημα 1. Ο ΚΗΠΟΣ

Υπάρχουν N δέντρα σε έναν κήπο. Το σχήμα του κήπου είναι τετράγωνο με μήκος πλευράς 1000m. Θέλουμε να βρούμε ένα ορθογώνιο παραλληλόγραμμο με το μεγαλύτερο εμβαδό το οποίο δεν περιέχει δέντρα. Οι πλευρές του παραλληλογράμμου πρέπει να είναι παράλληλες με τις πλευρές του τετραγώνου του κήπου. Οι πλευρές του παραλληλογράμμου μπορούν να περιέχουν δέντρα και μπορούν να εφάπτονται στις πλευρές του κήπου. Τα δέντρα δίνονται με συντεταγμένες (x,y) των θέσεών τους σε μέτρα. Τα δέντρα θεωρούνται σημεία χωρίς διαστάσεις.

Το (0,0) του συστήματος συντεταγμένων είναι η κάτω αριστερή πλευρά του κήπου και οι άξονες είναι παράλληλοι των πλευρών του.

INPUT DATA

Η πρώτη γραμμή του αρχείου INPUT.TXT περιέχει το N, τον αριθμό δηλαδή των δέντρων ($1 \leq N \leq 600$). Σε κάθε μία από τις επόμενες N γραμμές του αρχείου θα βρίσκονται δύο αριθμοί x και y που είναι οι συντεταγμένες των δέντρων ($0 < x < 1000, 0 < y < 1000$).

OUTPUT

Γράψτε στο αρχείο OUTPUT.TXT το εμβαδό του μεγαλύτερου παραλληλογράμμου.

ΠΑΡΑΔΕΙΓΜΑ

```
7
280 100
200 600
400 200
135 800
800 400
600 800
900 210
```

output

Πρόβλημα 2. ΤΟ ΠΑΡΤΥ

Ο πρόεδρος μιας εταιρείας οργανώνει ένα πάρτυ για τους εργαζόμενούς του. Η εταιρεία έχει μια ιεραρχική δομή. Οι σχέσεις προϊσταμένων διαμορφώνει ένα δέντρο του οποίου η "ρίζα" είναι ο ίδιος ο πρόεδρος. Για να κάνει το πάρτυ χαρούμενο για όλους, ο πρόεδρος θέλει να αποφύγει να καθίσουν μαζί στο ίδιο τραπέζι, ένας εργαζόμενος και ο προϊστάμενος του.

Το πρόβλημα είναι να υπολογίσετε τον ελάχιστο αριθμό τραπεζιών που χρειάζονται για να γίνει ένα πλάνο το οποίο να ικανοποιεί την παραπάνω συνθήκη. Η είσοδος στο πρόγραμμα είναι η άμεση σχέση προϊσταμένου. Ο P είναι προϊστάμενος του Q , αν ο P είναι άμεσα προϊστάμενος του Q ή υπάρχουν P_1, P_2, \dots, P_k ($1 < k$) άτομα έτσι ώστε $P=P_1$ και $Q=P_k$ και P_i είναι ο άμεσα προϊστάμενος του P_{i+1} ($i=1, \dots, k-1$).

Ο αριθμός των προσκεκλημένων είναι N ($1 \leq N \leq 200$) και υπάρχουν T θέσεις ($2 \leq T \leq 10$) σε κάθε τραπέζι.

ΕΙΣΟΔΟΣ

Στην πρώτη γραμμή του INPUT.TXT υπάρχει ο αριθμός των θέσεων ανά τραπέζι. Στη δεύτερη γραμμή ο αριθμός των προσκεκλημένων N . Στην επόμενη γραμμή, θα βρείτε N αριθμούς. Ο i αριθμός φανερώνει ποιός είναι ο άμεσα προϊστάμενος του i προσκεκλημένου. Ο πρόεδρος έχει τον αριθμό 1 και δεν έχει προϊστάμενο.

Π.χ.

```
4
13
0 1 9 9 9 2 2 1 1 7 8 8 10
```

ΕΞΟΔΟΣ

5

Πρόβλημα 3. ΤΑ ΠΟΤΗΡΙΑ ΜΕΤΡΗΣΗΣ

Σας δίνονται τρία ποτήρια. Ο όγκος του κάθε ποτηριού είναι 100cm^3 (κυβικά εκατοστά). Τα δύο πρώτα ποτήρια έχουν σημάδια, τα ίδια σε κάθε ποτήρι, και κάθε σημάδι γράφει τον όγκο του υγρού που περιέχεται μέσα στο ποτήρι μέχρι εκείνο το σημείο. Στην αρχή το πρώτο ποτήρι περιέχει 100cm^3 από κάποιο υγρό και τα άλλα δύο ποτήρια είναι άδεια. Γράψτε ένα πρόγραμμα το οποίο αποφασίζει αν μπορούμε να εξαγάγουμε στο τρίτο ποτήρι ποσότητα (όγκο) υγρού ίση με 1cm^3 , και αν μπορούμε τότε να δημιουργεί τα βήματα για να το κάνουμε. Σε κάθε βήμα, μετά τη λειτουργία του, πρέπει τουλάχιστον ένα από τα χρησιμοποιημένα ποτήρια να έχει υγρό μέχρι κάποιο σημάδι ή να είναι εντελώς άδειο. Καθ' όλη τη διάρκεια της διαδικασίας μπορούμε να κρατάμε την ακριβή ποσότητα του κάθε ποτηριού.

ΕΙΣΟΔΟΣ

Η πρώτη γραμμή του INPUT.TXT περιέχει το N , τον αριθμό των σημαδιών στα δύο πρώτα ποτήρια ($1 \leq N \leq 20$). Το υπόλοιπο αρχείο περιέχει N ακραίους σε αύξουσα σειρά που είναι τα νούμερα δίπλα στα σημάδια (οι όγκοι δηλαδή που μπορούν να μετρηθούν). Για κάθε σημάδι V ισχύει $1 \leq V \leq 100$.

ΕΞΟΔΟΣ

Γράψτε στο OUTPUT.TXT το string YES, αν μπορούμε να ξεχωρίσουμε όγκο ενός 1cm^3 στο τρίτο ποτήρι, ή NO αν δεν μπορούμε. Στην πρώτη περίπτωση γράψτε στη δεύτερη γραμμή του αρχείου τον αριθμό των βημάτων που απαιτούνται για να το κάνουμε.

ΠΑΡΑΔΕΙΓΜΑ

```
4
13 37 71 100
```

η έξοδος είναι:

```
YES
8
```

Day 1

Problem 1. ENCODING GRID (40 pts)

A kind of grid is sometimes used for encoding messages. Our naval fleet officials decide to use this kind of encoding for communication with their ship captains. The encoding grid is a square sheet of paper divided into $2N \times 2N$ little squares. N of little squares are cut out (see figure 1).

Let us briefly describe the encoding process. Take text of message of length $4N^2$. Then put an encoding grid onto a sheet of paper and begin to write first N^2 letters of message into the cut little squares (one letter per square, write letters all cut squares in the first line from left to right, then in the second line etc.). See example on figure 2 for "HELLOYELLOWWORLD". After finishing the first step rotate the encoding grid 90 degrees clockwise and continue similarly letters. After two more steps all letters should be written (see example on figure 3).

O###	H###	HOOY
##O#	##E#	LREO
O##O	L##L	LWEL
####	####	LLDW
(fig. 1)	(fig. 2)	(fig. 3)

It is necessary to have a correctly constructed grid. It means that when it is used in the way described above after each rotation new N^2 blank squares will appear under cut squares of the grid.

The naval fleet officials encoded thousands of messages with their grid and then they wanted to send them to ships. Unfortunately they lost the grid. Fortunately the naval admiral remembered the original text of one of the messages.

Your task is to get this original message text and encoded text and find one possible correctly constructed encoding grid with which the message text could have been encoded.

Input Data

The first line of input file contains an integer N ($1 \leq N \leq 10$). The second line contains $4N^2$ capital letters representing the original message. The next $2N$ lines contain the encoded text (each line $2N$ capital letters).

Output Data

The output file contains a correctly constructed grid with which given message could have been encoded. In each of $2N$ lines one line of grid ($2N$ characters) will be displayed where "O" (capital letter 'O' represents cut square and "#" uncut square.

Example

```

GRID.IN
2
HELLOYELLOWWORLD
HOOY
LREO
LWEL
LLDW

GRID.OUT
O###
##O#
O##O
####

```

Problem 2. SHIPS (30 pts)

The Palmia country is divided by a river into the north and south bank. There are N towns on both the north and south bank. Each town on the north bank has its unique friend town on the south bank. No two towns have the same friend.

Each pair of friend towns would like to have a ship line connecting them. They applied for permission to the government. Because it is often foggy on the river the government decided to prohibit intersection of ship lines (if two lines intersect there is a high probability of ship crash).

Your task is to write a program to help government officials decide to which ship lines they should grant permission to get maximum number of non intersecting ship lines.

Input data

The input file consists of several blocks. In the first line of each there are 2 integers X,Y separated with exactly one space, X represents the length of the Palmia river bank ($10 \leq X \leq 6000$), Y represents the width of the river ($10 \leq Y \leq 100$). The second line contains the integer N, the number of towns situated on both south and north riverbanks ($1 \leq N \leq 5000$). On each of the next N lines there are two non-negative integers C,D separated with space ($C,D \leq X$), representing distances of the pair of friend towns from the western border of Palmia measured along the riverbanks (C for the town on the north bank, D for the town on the south bank). There are no two towns on the same position on the same riverbank. After the last block there is a single line with two numbers 0 separated by a space.

Output Data

For each block of the input file the output file has to contain in consecutive lines the maximum possible number of ship lines satisfying the conditions above.

Example

SHIPS.IN

```
30 4
7
22 4
2 6
10 3
15 12
9 8
17 17
4 2
0 0
```

SHIPS.OUT

```
4
```

Problem 3. HIGHWAY TOLLS (30 pts)

In Palmia country they have N cities connected by highways (each highway connects exactly two cities in both directions). It is possible to reach every city from any other city using one or several highways. Till this year the highway tolls were collected on highways. In the middle of each highway there was a toll collector who collected 1 Palmia Coin (1 PC) from each car passing by.

Government officials decided to reduce the number of toll collectors and introduce highway stamps. If a car will want to enter a highway it must have this highway stamp placed on the window.

Officials decided that the highway stamp for one year will cost the same as 100 travels between two farthest cities. The distance between two cities is the minimum number of highways you need to use to get from the first city to the second one.

Your task is to write a program which computes the cost of the highway stamp.

Input Data

The input file consists of several blocks of data. Each block at the first line contains the integers N and M separated by a space where N is the number of cities and M the number of highways in Palmia country ($1 \leq N \leq 1000$, $1 \leq M \leq 2000$). Cities are numbered by integers from 1 to N. The next M lines contain each one pair of integers A,B ($1 \leq A,B \leq N$) separated by a space representing that there is a highway between the cities A and B. After the last block there is a single line with two numbers 0 separated by a space.

Output Data

For each block of input file the output file contains a single line with the cost of the highway stamp.

Example

TOLLS.IN

```
4 4
1 2
2 3
4 2
3 4
0 0
```

TOLLS.OUT

```
200
```

Day 2

Problem 1. BIN PACKING (40 points)

A factory has a bin packing robot at the end of an assembly line. There are exactly two bins open, and the robot packs items into any one of the open bins, one by one sequentially. The bins are identical, and one bin can accommodate a set of items up to a given weight limit. More precisely, the robot can perform the following four instructions:

1. Pack the current item into bin 1.
2. Pack the current item into bin 2.
3. Close bin 1 and open a new empty bin in place of the closed bin.
4. Close bin 2 and open a new empty bin in place of the closed bin.

A pack instruction can only be executed if the weight of the current item plus the sum of the weights of the items already in the bin does not exceed the limit. You are given the sequence of weights of items and a weight limit that is constant for all the bins. Write a program that computes the minimal number of bins needed by the robot to pack all items into them.

Input Data

The input file contains integer numbers. The first line of input file contains the weight limit L ($1 \leq L \leq 100$). This limit applies to all bins. The second line contains the number of items N ($1 \leq N \leq 5000$). Each of the following N lines contains the weight of an item. Each weight is at least 1 and at most L .

Output Data

Write on the first line of output file the minimum number of bins needed to pack all items.

Example

BINPACK.IN

8
6
4
2
5
3
5
4

BINPACK.OUT

3

Problem 2. CUTTING RECTANGLES (40 points)

You are given a rectangle whose side lengths are integer numbers. You want to cut the rectangle into the smallest number of squares, whose side lengths are also integer numbers. Cutting can be performed with a cutting machine that can cut only from side to side across, parallel with one side of the rectangle. Obtained rectangles are cut separately.

Input Data

The input file contains two positive integers in the first line: the lengths of the sides of the rectangle. Each side of the rectangle is at least 1 and at most 100.

Output Data

The output file consist of one line on which your program should write the number of squares resulting from an optimal cutting.

Example

CUTS.IN

5 6

CUTS.OUT

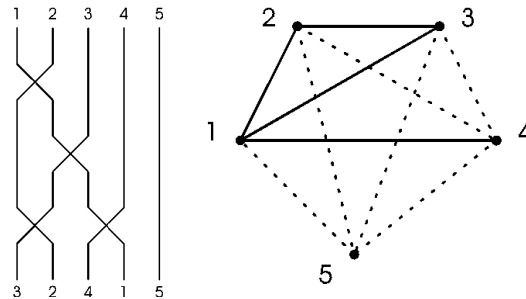
5

Problem 3. ELECTRICIAN (20 points)

The electrician Fero has just finished the installation of the new cable in the building of the CEOI secretariat. The cable consists of N wires and connects the computer room and CEOI secretary office. In the computer room Fero labeled the wires with numbers from 1 to N from left to right. Because the cable

was long, sometimes some of wires crossed between the computer room and the office (each pair of wires crosses at most once; at each moment only neighbouring wires may cross). So in the office the wires were not ordered the same way (see figure 1).

To know how these wires are crossed Fero drew the graph on the wall near the cable end in the computer room. The vertices represented wires and edges represented wire crosses. There is an edge between vertices a and b if and only if the wires a and b are crossed somewhere on the way (see figure 2 for graph representing situation in figure 1).



The cable does not function properly now. Unfortunately Fero broke his leg yesterday. We have another electrician but he is not so experienced in graph theory. He needs to know the order of wires at the end of the cable in the CEOI secretary office to correct this fault.

Input Data

In the input file there are several blocks of data. The first line of each block contains two numbers N and M separated by a space, where N ($1 \leq N \leq 100$) is the number of wires in the cable and M is the number of wire crosses. This is followed by M lines each containing pair of integers A, B separated by a space ($1 \leq A, B \leq N$) representing the cross of cable wires A and B somewhere on the way. After the last block there is a line with two numbers 0.

Output Data

For each block of the input file the output file contains a single line with the list of wire numbers in order as they appear at the second end of the cable (N numbers separated by space) or the message "IMPOSSIBLE" if no order is represented by given graph.

Example

ELECTRIC.IN

```
5 4
1 2
1 3
2 3
1 4
0 0
```

ELECTRIC.OUT

```
3 2 4 1 5
```

Problem 1. The Cave

There is a lot of caves in Byteland. This is the map of one of them:

All caves in Byteland have the following properties:

- All the chambers and passages are on the same level.
- No two passages cross each other.
- Some of the chambers are situated on the outer circle. They are called outer chambers.
- All the other chambers lie inside the outer circle and are called inner chambers.
- There is an entrance to the cave leading to one of the outer chambers.
- There are exactly three passages leading from every chamber to three different other chambers. If a chamber is an outer chamber then two of the passages lead to two neighbouring outer chambers on the circle and exactly one leads to an inner chamber. Passages connecting outer chambers are called outer passages and the remaining ones are called inner passages.
- It is possible to walk from one chamber to another using only inner passages but there is only one way to do that if we allow to walk through every inner passage at most once.
- Not all passages are equally hard to go through. They are divided into two groups: easy and hard.

It has been decided to open all the caves to the public. To assure a "smooth" and safe flow through a cave, visitors should follow a route marked in advance and visit every chamber of this cave exactly once. An exception from this rule is the entrance chamber where every tour begins and ends, you are allowed to visit this chamber exactly twice. The tour route should be fit for an average visitor and contain as few hard passages to walk through as possible.

Example

Consider the cave showed in the figure. The entrance chamber is 1. The dashed passages are hard to walk through. Route 1, 5, 4, 6, 8, 7, 2, 3 contains no hard passages at all. The final chamber, number 1, is implied and not listed.

Task

Write a program that:

- reads the description of a cave from the text file CAV.IN;
- finds a route through the cave that begins and ends in the entrance chamber, lets the visitors see all the other chambers only once and contains as few hard passages as possible;
- writes the result to the text file CAV.OUT

Input

There are two integers n , k (separated by a single space) in the first line of the text file CAV.IN

The integer n , $3 < n \leq 500$, is the number of all chambers in a cave and k is the number of all its outer chambers, $k > 3$. The chambers are numbered from 1 to n . Chamber 1 is the entrance chamber. Chambers 1, 2, ..., k are outer chambers. They do not have to appear on the outer circle in this order.

The next $3n / 2$ lines contain descriptions of the passages. Each description consists of three integers a, b, c separated by single spaces. The integers a and b are numbers of chambers at the ends of a passage. The integer c equals 0 or 1, where 0 means that the passage is easy to walk through and 1 that it is hard.

Output

Your program should write a sequence of n integers separated by single spaces to the first line of the text file CAV.OUT; the sequence has to begin with 1 (the entrance chamber).

The following $n - 1$ integers should be consecutive chambers on the computed route.

Example

For the text file CAV.IN:

```
8 5
1 3 0
3 2 0
7 3 1
7 2 0
8 7 0
1 8 0
6 8 0
6 4 0
6 5 1
```



```
5 4 0
2 4 0
5 1 0
```

one of the correct solutions is the text file CAV.OUT:
1 5 4 6 8 7 2 3

Time: 2 sec per each test

Problem 2. Hexadecimal Numbers

The base of the hexadecimal system is 16. In order to write down numbers in this system one uses 16 digits 0,1,...,9,A,B,C,D,E,F. The capital letters A,...,F stands for 10,...,15, respectively. For instance the value of the hexadecimal number CF2 is $12 * 16^2 + 15 * 16 + 2 = 3314$ in the decimal system.

Let X be the set of all positive integers whose hexadecimal representations have at most 8 digits and do not contain repetitions of any digit.

The most significant digit in such a representation is not zero.

The largest element in X is the number with the hexadecimal representation FEDCBA98, the second largest is the number FEDCBA97, the third is FEDCBA96 and so forth.

Task

Write a program that:

- reads positive integer n from the text file HEX.IN, n does not exceed the number of elements of X;
- finds the n-th largest element of X;
- writes the result to the text file HEX.OUT.

Input

The first line of the file HEX.IN contains integer n in the decimal representation.

Output

Your program should write the n-th largest element in X in the hexadecimal representation to the first line of the text file HEX.OUT

Example

For the text file HEX.IN:

```
11
```

the correct solution is the text file HEX.OUT:

```
FEDCBA87
```

Note: 2 sec for each test

Problem 3. Integer Intervals

An integer interval $[a,b]$, $a < b$, is a set of all consecutive integers beginning with a and ending with b.

Task

Write a program that:

- reads the number of intervals and their descriptions from the text file INT.IN;
- finds the minimal number of elements in a set containing at least two different integers from each interval;
- writes the result to the text file INT.OUT

Input

The first line of the text file INT.IN contains the number of intervals n, $1 \leq n \leq 10000$. Each of the following n lines contains two integers a, b separated by a single space, $0 \leq a < b \leq 10000$. They are the beginning and the end of an interval.

Output

Your program should write one integer to the first line of the text file INT.OUT; this should be the minimal number of elements in a set containing at least two different integers from each interval.

Example

For the text file INT.IN:

```
4
```

```
3 6
```

```
2 4
```

```
0 2
```

```
4 7
```

the correct solution is the text file INT.OUT:

```
4
```

Note: 2 sec for each test.

Problem 4. River Crossing

Citizens of Byteland adore all sports in which logical thinking is as important as physical skills. One of these sports is crossing the Hex River - the widest river in Byteland. There are n poles numbered $1..n$ (from left to right) stretching across the river. The citizens have to cross the river by going from the left bank to one pole perhaps to another and so on and then to the right bank.

The left bank is located one pole to the left of pole 1; the right bank is located one pole to the right of pole n .

At time 0, the citizen is on the left bank of the Hex River with the goal of reaching the right bank of the river as quickly as possible. At any given time, each pole is either up or down and the citizen is standing on a pole or standing on a bank of the river. A citizen can stand on a pole only if it is up; such a pole is available.

Each pole is down at time 0 and then spends a time units up, b time units down, a time units up, b time units down, etc. The constants a and b are defined separately for each pole. For example the pole with $a=2$ and $b=3$ will be down at time 0, up at time 1 and 2, down at time 3, 4, 5 and so on.

At time $t+1$, a citizen can choose to be on any available pole or river bank within 5 poles of his location at time t or even to stay on his current pole (if available) or bank. For example from pole 5 you can reach any of the poles 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or the left bank.

Task

Write a program that:

- reads the number of data blocks from the text file RIV.IN (each block contains a complete set of data for the problem);
- for each block
 - reads the number of poles and descriptions of their behaviour;
 - computes the first possible time the citizen can stand on the right bank, if it is reachable;
- writes the result to the text file RIV.OUT.

Input

The first line of the input file RIV.IN contains the number of data blocks x , $1 \leq x \leq 5$. The following lines comprise the x blocks. The first block starts on the second line of the input file; each subsequent block starts directly after the previous one.

The first line of each block contains an integer n , $5 < n \leq 1000$, the number of poles.

Each of the following n lines in the block contains two integers a, b separated by a single space, $1 \leq a, b \leq 5$. The integers in line $i + 1$ (in the block), $1 \leq i \leq n$, describe the behaviour of the pole i .

Output

For the k -th block, $1 \leq k \leq x$, write to the k -th line of the text file RIV.OUT one integer - the first time the citizen can reach the right bank or the word NO, when such a crossing is impossible.

Example

For the text file RIV.IN:

```
2
10
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
10
1 1
1 1
1 1
1 1
2 1
1 1
1 1
1 1
1 1
1 1
1 1
```

the correct solution is the text file RIV.OUT:
NO

Note: 2 sec for each test.

Problem 5. Shooting Contest

Welcome to the Annual Byteland Shooting Contest. Each competitor will shoot to a target which is a rectangular grid. The target consists of $r \times c$ squares located in r rows and c columns. The squares are coloured white or black. There are exactly two white squares and $r - 2$ black squares in each column. Rows are consecutively labelled $1, \dots, r$ from top to bottom and columns are labelled $1, \dots, c$ from left to right. The shooter has c shots.

A volley of c shots is correct if exactly one white square is hit in each column and there is no row without white square being hit. Help the shooter to find a correct volley of hits if such a volley exists.

Example

Consider the following target:

Volley of hits at white squares in rows 2, 3, 1, 4 in consecutive columns 1, 2, 3, 4 is correct.

Task

Write a program that:

- reads the number of data blocks from the text file SHO.IN; each block contains a complete set of data for the problem;
- for each block
 - reads the target size and the layout of white squares
 - verifies whether any correct volley of hits exists and if so, finds one of them
 - writes the result to the text file SHO.OUT;

Input

The first line of the input file SHO.IN contains the number of data blocks x , $1 \leq x \leq 5$. The following lines constitute x blocks. The first block starts in the second line of the input file; each next block starts directly after the previous one.

The first line of each block contains two integers r and c separated by a single space, $2 \leq r \leq c \leq 1000$. These are the numbers of rows and columns, respectively.

Each of the next c lines in the block contains two integers separated by a single space. The integers in the input line $i + 1$ in the block, $1 \leq i \leq c$, are labels of rows with white squares in the i -th column.

Output

For the i -th block, $1 \leq i \leq x$, your program should write to the i -th line of the file SHO.OUT either a sequence of c row labels (separated by single spaces) forming a correct volley of hits at white squares in consecutive columns 1, 2, ..., c , or one word NO if such a volley does not exist.

Example

For the text file SHO.IN:

```
2
4 4
2 4
3 4
1 3
1 4
5 5
1 5
2 4
3 4
2 4
2 3
one of the correct solutions is the text file SHO.OUT:
2 3 1 4
NO
```

Note; 2 sec for each test.

Problem 6. Dominoes

A domino is a flat, thumbsized tile, the face of which is divided into two squares, each left blank or bearing from one to six dots. There is a row of dominoes laid out on a table:

6	1	1	1
1	5	3	2

The number of dots in the top line is $6+1+1+1=9$ and the number of dots in the bottom line is $1+5+3+2=11$. The gap between the top line and the bottom line is 2. The gap is the absolute value of difference between two sums. Each domino can be turned by 180 degrees keeping its face always

upwards. What is the smallest number of turns needed to minimise the gap between the top line and the bottom line?

For the figure above it is sufficient to turn the last domino in the row in order to decrease the gap to 0. In this case the answer is 1.

Task

Write a program that:

- reads the number of dominoes and their descriptions from the text file DOM.IN,
- computes the smallest number of turns needed to minimise the gap between the top line and the bottom line,
- writes the result to the text file DOM.OUT.

Input

The first line of the text file DOM.IN contains an integer n , $1 \leq n \leq 1000$. This is the number of dominoes laid out on the table.

Each of the next n lines contains two integers a, b separated by a single space, $0 \leq a, b \leq 6$. The integers a and b written in the line $i+1$ of the input file, $1 \leq i \leq 1000$, are the numbers of dots on the i -th domino in the row, respectively, in the top line and in the bottom one.

Output

Your program should write exactly one integer to the first line of the text file DOM.OUT; this integer should be the smallest number of turns needed to minimise the gap between the top line and the bottom line.

Example

For the text file DOM.IN:

```
4
6 1
1 5
1 3
1 2
the correct solution is the text file DOM.OUT:
1
```

Note: 2 sec per test

..

American Computer Science League

'97 All-Star Contest Programming Problems

May 24, 1997

East HS, Salt Lake City, Utah

This page contains the programming problems that were given at the 1997 ACSL All-Star Contest. Test data and sample solutions (from those who competed) will be posted on June 30th.

Program 1: Time Card (Junior Division, 5 points)

Jenny just started work as a programmer for Justine's Java Workshop. She is paid \$10 an hour, with a few exceptions. She earns an extra \$1.50 an hour for any part of a day where she works more than 8 hours, and an extra \$2.50 an hour for hours beyond 40 in any one week. Also, she earns a 125% bonus for working on Saturday, and a 50% bonus for working on Sunday. The bonuses for Saturday and Sunday are computed based on the hours worked those days; they are not used to calculate any bonus for working more than 40 hours in a week.

You'll be given the number of hours Jenny worked each day in a week (Sunday, Monday, ..., Saturday), and you need to compute her salary for the week. The input will be positive integers, less than or equal to 24. The output must be formatted with a dollar sign and rounded to the nearest penny. For example, "\$2" and "\$2.136666" are wrong answers; the correct versions are "\$2.00" and "\$2.14", respectively. There may not be any embedded spaces in your answers. There will be 5 sets of data.

Sample Input:

```
Line 1: 0, 8, 8, 8, 10, 6, 0
Line 2: 4, 0, 0, 0, 0, 6, 0
```

Sample Output:

```
Output 1: $403.00
Output 2: $120.00
```

Program 2: Botchagaloop (Junior Division, 5 points)

The botchagaloop value of a number x is found as follows. First, convert x to base 8. Call this p . Next, sort the digits of p in increasing order. Call this q . Subtract q from p (in base 8, of course). Repeat the "sort-subtract" sequence 4 more times, or until the digits in the result are in sorted order (whichever come first). Finally, convert the number back to base 10.

For example, 3418 has a botchagaloop value of 1008. It is computed as follows: $3418 = 6532$ (base 8); $6532 - 2356 = 4154$; $4154 - 1445 = 2507$; $2507 - 257 = 2230$; $2230 - 223 = 2005$; $2005 - 25 = 1760$; and finally, $1760 = 1008$ (base 10). Note that there is at least one subtraction and at most 5 subtractions.

There will be 5 inputs. Each is a positive integer less than 1,000,000. Print the botchagaloop value of each input.

Sample Input:

```
Line 1: 3418
Line 2: 123
```

Sample Output:

```
Output 1: 1008
Output 2: 28
```

Program 3: Digit Count (Juionr and Intermediate Divisions, 10 points)

Consider those 5 digit numbers (i.e., between 10,000 and 99,999 inclusive) that can be written in the form $(a^{**n})(b^{**m})$ where a and b are distinct prime numbers, and n , and m are non-negative integers. How many times does the digit k appear?

For example, there are four 5-digit numbers whose only prime factors are 13 and 19: 28561, 41743, 61009, and 89167. The digit zero appears twice, the digit one appears 4 times, and so on.

There will be 10 inputs, each consisting of three positive integers, a , b , and k , in that order.

Sample Input:

```
Line 1: 5, 13, 6
```

Line 2: 11, 37, 2

Sample Output:

Output 1: 3
Output 2: 1

Program 4: You Are El (Junior, Intermediate, Senior Divisions, 10 pts)

Pages on the World Wide Web are accessed by a unique identifier called a URL (Uniform Resource Locator). For example, the URL of the file that you are reading right now is

<http://www.acsl.org/contests/96/ALLSTAR/prog4.ps>

There are three primary parts to a URL: protocol, host, and path. The protocol, http in this case, specifies how the file will be transferred from one computer to another. Other common protocols are ftp, gopher, and file. The host is the name of the computer on which the Web page resides. The ACSL Web pages all reside on the machine called www.acsl.org.

Finally, the path (contests/96/ALLSTAR/prog4.ps) indicates where on the host computer the file is located.

The syntax of a URL is as follows: The protocol is the part of the URL up to the colon. The host is the part of the URL following the colon-slash-slash. It ends at a slash. The path starts after the slash ending the host. The path is optional; if it's missing, assume the path is default.htm. Also, if the path ends with a slash, append default.htm to it.

Links on a Web page p can be specified in one of three ways: in absolute terms, relative to the server hosting p, or relative to page p.

An absolute URL starts with a protocol (e.g., <http://www.cnn/>). This is used to jump to any arbitrary page on the Web. For example, if this page (the one you are reading right now!) contained the link <http://www.microsoft.com/>, clicking on it would cause you to jump to the Microsoft home page.

A relative-to-the-server URL starts with a slash (e.g., </contests/95/SR/shorts3.ps>). This says to look for the page on the same server as p.

Finally, all other URLs are assumed to be relative to the page on which they reside. For example, the link [prog5.ps](#) on this page refers to the URL

<http://www.acsl.org/contests/96/ALLSTAR/prog5.ps>

The link [../INT/short2.ps](#) refers to <http://www.acsl.org/contests/96/INT/short2.ps>. That is, two dots indicates to go up a directory in the path. To make you life easy (see, we're nice guys!), the two dots will only appear at the start of a path. Of course, it might appear a few times (e.g., [../../94/JR/short1.ps](#)).

In this program, you'll be given the URL of a Web page and a link on the page, and you need to find the URL that the link will jump you to.

Sample Input:

Line 1: <http://www.acsl.org/contests/96/ALLSTAR/shorts.ps>, programs.ps
Line 2: <http://www.acsl.org:8000/SampleContests/95/ALLSTAR/shorts.ps>, /flyer.ps
Line 3: <http://www.acsl.org/SampleContests/95/ALLSTAR/shorts.ps>, <http://www.cnn/>

Sample Output:

Output 1: <http://www.acsl.org/contests/96/ALLSTAR/programs.ps>
Output 2: <http://www.acsl.org:8000/flyer.ps>
Output 3: <http://www.cnn/default.htm>

Program 5: A Balancing Act, Revisited (Intermediate and Senior Divisions, 10 points)

In the last contest, you explored an approach to balancing a binary search tree: you maintained multiple binary search trees and as each new key was inserted, you inserted it into the tree where it would be closer to the tree's root. This problem tries a different approach at maintaining multiple trees.

The idea here is to consider the trees from left to right, and insert the new key into the first tree where it will be less than or equal to a specified depth k . If no such trees exist, start a new leftmost tree.

For example, inserting the string AMERICANA into trees of depth 1 or less will cause 3 trees to be created, whose roots are I, E, and A (in that order):



The input will be 10 sets of data. Each set will consist of a number k and a string s (up to 128 characters long). In the string s , ignore everything but the letters A through Z; uppercase and lowercase are the same. For each input pair, build the trees to depth k with the letters in s as described above. Print the root nodes of the trees in order from left to right.

Sample Input:

Line 1: 3, AMERICAN COMPUTER SCIENCE
Line 2: 4, SALT LAKE CITY, UTAH

Sample Output:

Output 1: E R C A
Output 2: E S

Program 6: The Biggest, Revisited (Intermediate and Senior Divisions, 10 points)

You'd have thought that after Contest #1, you'd never hear from the Mighty Modulas again. But you'd be wrong. The Mighty Modulas have been at it again: They've been given index cards printed with numbers on both sides, and they've torn each card between the digits. Your job, as virtual leader of the Mighty Modulas, is to teach your charges how to tear each card so that the numbers on all the scraps add up as close as possible to some given target number without going over. You are free, of course, to turn over any scrap.

Consider, for example, a card with the number 534 on one side and 197 on the other side. There are 3 ways that you could tear the card: between the 5 and the 3; between the 3 and the 4; and after the 5 and after the 3. In the first case, you have two strips: one with 5 on one side and 7 on the other, and the other strip with 34 on one side and 19 on the other. In the second case, you'd also have two strips: one with 53 on one side and 97 on the other, and the second strip with 4 on one side and 1 on the other. Finally, in the third case, you'd have 3 strips. You must make at least one tear. If the numbers are different lengths, pad the shorter one with leading zeros so that they are of the same length.

You'll be given 10 sets of data. Each set consists of three numbers: the number on one side of the card, the number on other side of the card, and a target number. You are to figure out how to tear the card such that the numbers on one side or the other of the torn scraps add up as close as possible to the target number without going over. Print out the sum. The number on the cards will be positive integers less than 1,000,000,000.

Sample Input:

Line 1: 534, 197, 50
Line 2: 534, 197, 100
Line 2: 251, 8, 60

Sample Output:

Output 1: 41
Output 2: 98
Output 3: 59

Program 7: Squarepoint (Intermediate and Senior Divisions, 10 points)

Given a rectangle and a point, report the distance from the point to the nearest point on the perimeter of the rectangle.

There will be ten sets of data. Each set of data consists of the two corners of a rectangle, followed by coordinates (x,y) of a point p . To make input easy on the proctor, the corners of the rectangle will be a character that refers to one of the following 9 points:

A (2, 3) B (11, 6) C (7,11)

D (1,10) E (5,16) F (5, 6)
G (0,40) H (9, 9) I (7, 0)

For each input, print the distance from p to the closest spot on the perimeter of the rectangle. Your answers must be within 0.01 of our answers.

Sample Input:

Line 1: FH 7,8
Line 2: CB 5,9

Sample Output:

Output 1: 1
Output 2: 2

Program 8: Alphaddition (Senior Division, 10 points)

Remember those "brain puzzlers" of the form:

$$ABCD + BDC = EAEA$$

In case you don't remember them, the problem is pretty straightforward: find numbers to replace the A, B, C, ... that make the equation true. The replacements must be consistent; that is, replace all A's by one digit, all B's by a different digit, and so on. For example, the equation $1538 + 583 = 2121$ solves the equation above. The equation has 5 more solutions: $1547 + 574 = 2121$, $1574 + 547 = 2121$, $1583 + 538 = 2121$, $3658 + 685 = 4343$, and $3685 + 658 = 4343$. The second sample problem below has three solutions: $546+54=600$, $576+57=633$, and $586+58=644$.

There will be 10 inputs. Each input consists of three strings, p, q, and r. The strings are composed of the letters A through Z and numbers. You need to report the number of solutions there are to the problem $p+q=r$. The leading digit in each term cannot be a 0. Also, to keep the running time of this program reasonable, we promise that there won't be more than 4 different letters in the original data.

Sample Input:

Line 1: AA + BC = 100
Line 2: XYZ + XY = 6QQ

Sample Output:

Output 1: 7
Output 2: 3

All Ireland School Programming Competition 1994

Round 1

Problem 1. Horizontal Histogram

Write a program that accepts a set of digits (0 to 9) as input and prints a horizontal histogram representing the occurrences of each digit.

Test your program with the set of 13 digits:
1, 7, 2, 9, 6, 7, 1, 3, 7, 5, 7, 9, 0

Example

Enter a Number : 12
Enter 12 digits:
1,7,2,9,6,7,1,3,7,5,7,9

```
0
1 **
2 *
3 *
4
5 *
6 *
7 *****
8
9 **
```

Problem 2. Vertical Histogram

Write a program that accepts a set of digits (0 to 9) as input and prints a vertical histogram representing the occurrences of each digit.

Test your program with the set of 13 digits:
1, 7, 2, 9, 6, 7, 1, 3, 7, 5, 7, 9, 0

Example

Enter a Number : 12
Enter 12 digits:
1,7,2,9,6,7,1,3,7,5,7,9

```
      *
      *
*    * *
*** ** *
0123456789
```

Problem 3. Well Ordered Numbers

The number 138 is called well-ordered because the digits in the number (1,3,8) increase from left to right ($1 < 3 < 8$). The number 365 is not well-ordered because 6 is larger than 5.

Write a program that will find and display all possible three digit well-ordered numbers. Report the total number of three digit well-ordered numbers.

Example

The three digit well ordered numbers are:
123 124 125 126 127 128 129 134

135 136 137 138 139 145 146 147
 148 149 156 157 158 159 167 168

 678 679 689 789

The total number is ??

Problem 4. Roman Numerals

Write a program that reads in a roman numeral and converts the number to normal decimal. The roman characters are :

M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, I = 1

Test your program with the following values : V (5), IV (4), VIII (8), MM (2000), MCM (1900), MCMXCIV (1994).

Example

Enter a roman numeral : VIII
 VIII is 8.

Problem 5. Hotel Keys

The logo hotel gives out cards instead of keys. Each card contains a unique combination of holes, and can open one door in the hotel. The lock works by shining a beam of light through the card and detecting the pattern made by the holes. If the pattern is correct then the door opens. Each guest will be given a card, that opens only his or her door. An example card is shown here :

Each card has 4 x 6 (24) locations where a hole could go. This example has six holes. The holes of a card are not allowed to form a complete row, column or diagonal as otherwise, the card would easily bend or tear. (A diagonal can have two, three or four holes.) There must be at least five holes in each card.

No card should be able to open any door other than its own, no matter which way it is turned.

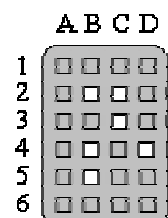
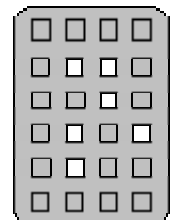
Write a program that will work out patterns for a number of cards. You don't have to draw the card, simply print out the locations of the holes for each card. For example :

The holes on this card are at B2, C2, C3, B4, D4, and B5. Note that another hole at A1 would not be allowed as then a diagonal is formed across the card. (A diagonal can consist of two holes, so holes at D5 and C6 would also be disallowed.)

Get your program to produce 20 cards.

Example

Enter the number of cards required : 2
 Card 1 : Holes at A1, B1, C1, B2, C2.
 Card 2 : Holes at B1, C1, D1, B2, B3, B4.



Your program should be called day1a.bas if it is a BASIC program, day1a.pas if it is a Pascal program, day1a.c if it is a C program, day1a.cpp if it is a C++ program and day1a.lg if it is a logo program.

It should read its input from a file called day1a.dat, and write its output to a file called day1a.sol. The two example input files and the solutions are on the D: drive. The names of the input files are test1a1 and test1a2. The solution files are called test1a1.sol and test1a2.sol.

Note that if your program generates correct results for these files, it does not mean that your program is correct.

N.B. If you are writing a BASIC program ensure that it terminates with a system command, so that it exits to DOS. This is essential if it is to be processed automatically. You will get zero marks for the program if you neglect to do this.

Problem 2. Factorials

The factorial of an integer n , written $n!$, is the product of all the integers from 1 to n inclusive. The factorial quickly becomes very large; $13!$ is too large to store as an integer on most computers, and $35!$ is too large for a floating-point variable. Your task is to find the rightmost non-zero digit of $n!$. For example, $5! = 1 * 2 * 3 * 4 * 5 = 120$, so the rightmost non-zero digit of $5!$ is 2. Also, $7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040$, so the rightmost non-zero digit of $7!$ is 4.

Mike Scott provided some more information about factorials

Input

An integer n , between 1 and 100 inclusive.

Output

The rightmost non-zero digit of $n!$.

Example 1

Input
3
Output
6

Example 2

Input
10
Output
8

Instructions

Your program should be called day1b.bas if it is a BASIC program, day1b.pas if it is a Pascal program, day1b.c if it is a C program, day1b.cpp if it is a C++ program and day1b.lg if it is a logo program.

It should read its input from a file called day1b.dat, and write its output to a file called day1b.sol. The two example input files and the solutions are on the D: drive. The names of the input files are test1b1 and test1b2. The solution files are called test1b1.sol and test1b2.sol.

Note that if your program generates correct results for these files, it does not mean that your program is correct.

N.B. If you are writing a BASIC program ensure that it terminates with a system command, so that it exits to DOS. This is essential if it is to be processed automatically. You will get zero marks for the program if you neglect to do this.

Problem 3. Transformations

A square pattern of black and white tiles is transformed into another square. Write a program that will recognise the minimum transformation that has been applied to the original pattern given the following list of possible transformations:

- 90 Degree Rotation: The pattern was rotated to the right 90 degrees.
- 180 Degree Rotation: The pattern was rotated to the right 180 degrees.
- 270 Degree Rotation: The pattern was rotated to the right 270 degrees.
- Vertical Reflection: The pattern was reflected vertically. (See example 4.)
- Combination: The pattern was reflected vertically and then subjected to one of the rotations.
- No Change: The original pattern was not changed.
- Invalid Transformation: The new pattern was not obtained by any of the above methods.

Input

The input file will consist of a number n (between 1 and 10 inclusive), the size of the square, followed by n lines of the original pattern, and then after a blank line, the n lines of the transformed pattern. Black squares will be indicated by a '.', and white squares by an 'X'.

Output

The output from your program will be a phrase describing the transformation that changed the original pattern to the new pattern. If more than one transformation is possible, then your program should show the transformation corresponding to the minimal amount of work necessary to convert the original pattern to the new pattern. For the purposes of evaluating the amount of work needed, rotations are considered less work than reflections, and smaller rotations are less work than larger ones.

Note that only the above possibilities should be considered - there is no such thing as a 360 Rotation, nor is there a horizontal reflection. Also remember that if a single rotation is not sufficient, your program should consider a reflection followed by a rotation.

Example 1 Input 5 X...X .X... ..X. ..X.XX X ...X. .X... ..X.. XX..X Output ROTATED 90 DEGREES.	Example 2 Input 6 ...XX ...X.. XX..X. ..X... ...X.. ..X..X X...X X.X... .X..X. ...X.X ..X... ..X... Output ROTATED 270 DEGREES.	Example 3 Input 2 X. .X X. .X Output NOT TRANSFORMED.
Example 4 Input 4 ..X. XX..X ...X XX.. ..X. Output REFLECTED.	Example 5 Input 5 X.... .X... .X... ...X.X .X... ..X.. ..X..X X.... Output IMPROPER TRANSFORMATION.	Example 6 Input 4 .X.. .X.XX. ..X. X... ..XX Output REFLECTED AND ROTATED 270 DEGREES

Instructions

Your program should be called day1c.bas if it is a BASIC program, day1c.pas if it is a Pascal program, day1c.c if it is a C program, day1c.cpp if it is a C++ program and day1c.lg if it is a logo program.

It should read its input from a file called day1c.dat, and write its output to a file called day1c.sol. The two example input files and the solutions are on the D: drive. The names of the input files are test1c1 and test1c2. The solution files are called test1c1.sol and test1c2.sol.

Note that if your program generates correct results for these files, it does not mean that your program is correct.

N.B. If you are writing a BASIC program ensure that it terminates with a system command, so that it exits to DOS. This is essential if it is to be processed automatically. You will get zero marks for the program if you neglect to do this.

Day 2

There are three problems contained in this document. The first problem day2a is the main problem for the day. The other two problems (day2b and day2c) are the Tie Break problems 1 and 2.

Note there are many more marks going for the main problem than there are for Tie Break problems. You should only do the TieBreak problem if you feel that you can't improve on the main problem.

With each problem there will be a number of examples. For the main problem, some of the examples will be used (in addition to other unseen tests) to test your program. In the Tie Break problems all the tests will be unseen. The example files and solutions will be on the hard disk. The names of these files will be as given in the problem.

When you are finished, ensure that you have saved your files onto the root directory of the D: drive. (That is, a BASIC program for problem A should be saved as D:\day2a.bas.) You will also be given a floppy disk at the end, and you must also save your files to this disk and give it to a supervisor.

Please note that you cannot write to the C: drive.

Best of luck

Problem 1. Raucous Rockers

You just inherited the rights to n previously unreleased songs recorded by the popular group Raucous Rockers. You plan to release a set of m compact disks with a selection of these songs. Each disk can hold a maximum of t minutes of music, and a song can not overlap from one disk to another.

Since you are a classical music fan and have no way to judge the artistic merits of these songs, you decide on the following criteria for making the selection:

- a) The songs will be recorded on the set of disks in the order of the dates that they were written.
- b) The total number of songs included will be maximized.

Input

The first line contains the values of n , t and m (integer numbers) followed by a line containing a list of the lengths of n songs, ordered by the date they were written. No song will be too long to fit on a disk, and it will not be possible to put all the songs on the disks

Output

The output will be an integer indicating the number of songs that, following the above selection criteria will fit on m disks.

Example 1

Input

```
10 5 3
5, 5, 5, 5, 5, 5, 5, 5, 5, 5
```

Output

```
3
```

In this example there are ten songs, and three disks. Each disk can hold 5 minutes worth of songs. In this case, all the songs last five minutes, so obviously only three songs can be recorded.

Example 2

Input

```
5 6 4
4, 3, 4, 4, 5
```

Output

```
4
```

Here there are 5 songs and 4 disks that can hold 6 minutes each. However it is not possible to hold more than one song on any disk, so only four songs can be recorded.

Example 3

Input

```
10 5 3
3, 5, 1, 2, 3, 5, 4, 1, 1, 5
```

Output

```
6
```

In this example there are ten songs, and three disks. The second line contains the lengths of each of the songs. E.G. the first song that was recorded lasts three minutes. The maximum number of songs that can fit on one disk is 6. (Remember that they have to be in date order.)

Problem 2. Greedy Gift Givers

You are to determine, for a group of gift-giving friends, how much more each person gives than they receive (and vice versa for those that view gift-giving with cynicism). In this problem each person sets aside some money for gift-giving and divides this money evenly among all those to whom gifts are given. However, in any group of friends, some people are more giving than others (or at least may have more acquaintances) and some people have more money than others.

Given a group of friends, the money each person in the group spends on gifts, and a (sub)list of friends to whom each person gives gifts; you are to write a program that determines how much more (or less) each person in the group gives than they receive.

Input

The input is a group of gift-givers which consists of several lines:

the number of people in the group,

a list of the names of each person in the group,

a line for each person in the group consisting of the name of the person, the amount of money spent on gifts, the number of people to whom gifts are given, and the names of those to whom gifts are given.

All names are lower-case letters, there are no more than 10 people in a group, and no name is more than 12 characters in length. Money is a non-negative integer less than 2000.

Output

The name of each person in the group should be printed on a line followed by the net gain (or loss) received (or spent) by the person. Names in a group should be printed in the same order in which they first appear in the input.

All gifts are integers. Each person gives the same integer amount of money to each friend to whom any money is given, and gives as much as possible. Any money not given is kept and is part of a person's "net worth" printed in the output.

Example 1

Input

```
5
dave laura owen vick amr
dave 200 3 laura owen vick
owen 500 1 dave
amr 150 2 vick owen
laura 0 2 amr vick
vick 0 0
```

Output

```
dave 302
laura 66
owen -359
vick 141
amr -150
```

Example 2

Input

```
3
liz steve dave
liz 30 1 steve
steve 55 2 liz dave
dave 0 2 steve liz
```

Output

```
liz -3
steve -24
dave 27
```

Problem 3. The Cat in the Hat

A clever cat walks into a messy room which he needs to clean. Instead of doing the work alone, it decides to have its helper cats do the work. It keeps its (smaller) helper cats inside its hat. Each helper cat also has helper cats in its own hat, and so on.

Eventually, the cats reach a smallest size. These smallest cats have no additional cats in their hats. These unfortunate smallest cats have to do the cleaning.

The number of cats inside each (non-smallest) cat's hat is a constant, n . The height of these cats-in-a-hat is $1/(n+1)$ times the height of the cat whose hat they are in.

All heights are positive integers.

Given the height of the initial cat and the number of worker cats (of height one), find the number of cats that are not doing any work (cats of height greater than one) and also determine the sum of all the cats' heights (the height of a stack of all cats standing one on top of another).

Input

The input is a single line consisting of two positive integers, separated by white space. The first integer is the height of the initial cat, and the second integer is the number of worker cats.

Output

Print the number of cats that are not working, followed by a space, followed by the height of the stack of cats.

Example 1**Input**

216 125

Output

31 671

Example 2**Input**

5764801 1679616

Output

335923 30275911

All Ireland Schools Programming Competition 1995

Round 1

Problem 1. Factorials

The factorial of an integer n , written $n!$, is the product of all the integers from 1 to n inclusive. For example, $5! = 1 * 2 * 3 * 4 * 5 = 120$. Write a program that accepts a number, n ($1 \leq n \leq 12$), as input and prints the factorial of that number.

Example

```
Enter a Number : 4
4! = 24
```

Test your program with the following numbers : 1, 4, 6, 10, 12.

Problem 2. Is this a Triangle

Your job is to classify a triangle based on the lengths of its sides. Write a program to take three numbers, a , b and c , which are the lengths of the sides of a triangle, and classify the triangle as isosceles, equilateral, possible or impossible.

Note that an isosceles triangle has two equal sides, an equilateral triangle has three equal sides, an impossible triangle cannot be formed from the three sides (e.g. $a = 1$, $b = 2$, and $c = 5$) and a possible triangle is everything else.

Example

```
Enter three sides : 2 4 3
possible
```

Test your program with the following sides

```
2 2 2
1 2 5
1 1 2
2 3 4
```

Problem 3. Common Letters

Write a program that takes two words and finds any common letters that they have. For example, the words 'computer' and 'program' have the letters 'o', 'm', 'p', 'r' in common. The output should be both words with all common letters in capitals. Neither word will have more than 8 letters.

Example

```
Enter two words : computer program
cOMPuTeR PRoGrAm
```

Test your program with the following pairs of words

```
hello all
bye all
```

4. Word Chain

A word chain is a daisy chain made from words. Two words, w_1 and w_2 , may be linked if the last letter of w_1 is the same as the first of w_2 . E.g. 'bar' and 'red' may be linked. You must form a word-chain from a list of words that you are given. The word-chain must

- consist of four words

- be circular (i.e. the last word should link to the first word)

No word may be used more than once.

Input

The input will be a list of N words. The first item will be N , ($4 \leq N \leq 25$), followed by the list of words. No word will have more than 8 letters.

Output

Will be a list of four words that form a chain.

Example

```
Enter the list of words : 5
gap pet big tab tar
A four word chain is : big-gap-pet-tab
```

Note

'tar' couldn't replace tab because then it wouldn't link back to 'big'.
This is the only possible chain. Because it is circular, it could also be written as gap-pet-tab-big.

Test your program with the following list

```
22
say how many south marks may be formed dub yam for yas sos biff
deaf by da why sam saw sob ross
```

For additional marks, say how many different four word chains may be formed.

Problem 5. Number Triangle

```
  7
 3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

The above is a number triangle. Write a program that calculates the highest sum of numbers passed on a route that starts at the top and ends somewhere on the base. You may only move downwards to one of the two numbers diagonally below.

The number of rows in the triangle is > 1 but ≤ 100 .
The numbers in the triangle, all integers, are between 0 and 99.

The route in this case is shown in bold: 7, 3, 8, 7 and 5. The sum is 30.

The input will consist of N, the number of rows, followed by N rows of digits that make up the triangle. The output is the highest sum that is reached.

Example

```
Enter a number triangle :
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
The solution is 30.
```

Test your program with the above triangle and the one below

```
4
1
1 2
1 1 3
5 3 0 1
```

All Ireland Schools Programming Competition 1996

Round 1

1. Maximin

Write a program that will find the maximum and the minimum of a list of positive numbers.

Example

```
How many numbers ? 8
4 2 7 23 10 9 19 5
The maximum is 23, the minimum is 2.
```

2. Egg Timer

Write a program that will ask the user for a whole number N between 3 and 10 and print an egg timer of size N.

Example

```
Enter a number ? 4
*_*_*_*
*_*_*
*_*
*
*
*_*
*_*_*
*_*_*_*
```

Problem 3. Remainder

You are given two words, and your job is to remove all letters from the first word that are contained in the second. For example, if you are given the words "computer program", remove letters from the word "computer" if they occur in "program", leaving the letters "cute".

Example

```
Enter two words : computer program
Remainder : cute
```

Problem 4. Encryption

A secret code reorders the letters in a message using an array. E.g. the following message "Attack at dawn or we are done for" is placed in a 6*6 array :

```
Attack
*at*da
wn*or*
we*are
*done*
for...
```

Blanks are replaced by asterisks, and full stops are added to the message so that it fills out the array. The coded message is obtained by reading down the columns, i.e.

```
A*ww*ftanedott**ora*oan.cdrre.ka*e*.
```

Write a program that will encode a message as above. Use an array with 6 columns, and adjust the number of rows depending on the message length.

Example

```
Enter a message : Attack at dawn or we are done for
The encrypted message is : A*ww*ftanedott**ora*oan.cdrre.ka*e*.
```

For extra marks, write a program to decode a message that has been encrypted using the above rules.

Problem 5. Come to Nothing

Consider the sequence of digits from 1 through N (where $N \leq 9$) in increasing order:

```
1 2 3 4 5 ....N
```

Insert either a + (for addition) or a - (for subtraction) between each of the digits so that the resultant sum is zero. Print all possible combinations that sum to zero.

Example

```

Enter a number ? 7
1 + 2 - 3 + 4 - 5 - 6 + 7 = 0
1 + 2 - 3 - 4 + 5 + 6 - 7 = 0
1 - 2 + 3 + 4 - 5 + 6 - 7 = 0
1 - 2 - 3 - 4 - 5 + 6 + 7 = 0

```

Final Round

Day 1

Problem 1. The In between Sum

You are given two numbers, A and B, and you have to find the total of all the numbers between these two numbers. For example if you are given the numbers 9 and 15, you must find the sum 10+11+12+13+14 which is 60.

Input

The input is the pair of integers, A and B ($1 \leq A \leq B \leq 30000$).

Output

The output will be the sum of all the numbers beteen A and B.

Example 1 Input 9 15 Output 60	Example 2 Input 10 20 Output 135
---	---

Problem 2. Word Construction

A word (doublet) is written on two cards, for example :

COMPUTER COMPUTER

A new word is formed by joining a substring from the first card to the second, e.g. :

PUT TER

Here the word "PUTTER" is formed.

You will be given one doublet, followed by a list of words, and you must print all words from the list that can be formed in the above way as a pair of substrings.

For this problem, valid substrings must have at least two letters.

Input

The input is the doublet, followed by N, ($1 \leq N \leq \text{THE } N \text{ words}$), one per line. All words (including the doublet) will be in uppercase letters, and will have between 2 and 32 letters.

Output

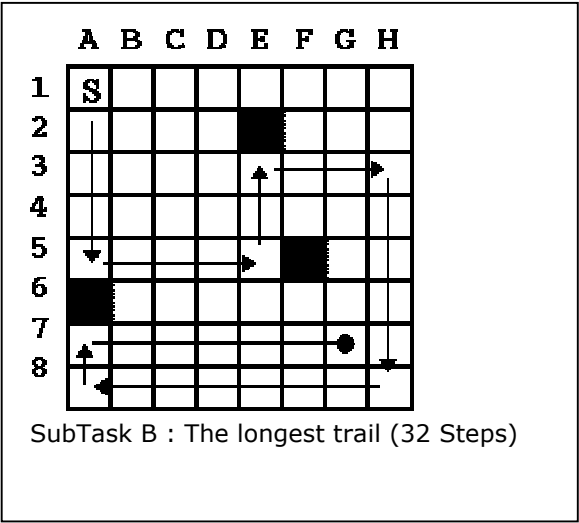
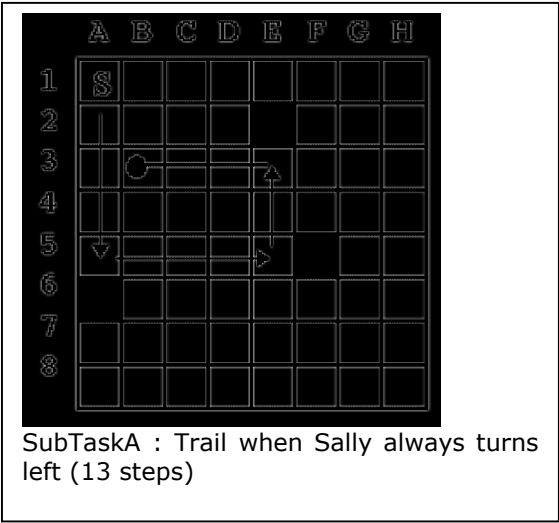
The output will be any words that can be formed by taking two substrings from the doublet. Each word should appear on a line on its own.

Example 1 Input COMPUTER 3 ONION CUTE PUTTER Output PUTTER	Example 2 Input PAW 3 APAW PAPA WAP Output PAPA
---	--

Problem 3. Snail Trail

Sally the Snail lives on an 8x8 grid which contains some obstructions (shown below as black squares). Sally always moves as far as she can in a straight line. When she reaches an obstruction or the edge of the board, she turns either right or left. When she encounters a square she has already visited, she stops.

Sally always starts at square A1, and begins moving in a downward direction. You will be given the location of all the black squares on the grid, and your task is to find out how many steps Sally can take before getting blocked. In subtask A, Sally always turns to her left, and in subtask B, Sally turns either left or right to maximise the length of the trail.



The diagrams show an example grid with three black squares (at A6, E2 and F5). The trails for each subtask are also shown. In subtask A, Sally moves from A1 down to A5, where she turns to her left and heads off to E5. From here, she turns left to square E3, and again turns left to finish at B3. She has taken 13 steps.

In subtask B, Sally wants the longest route, which is shown in the diagram to have 32 steps.

Input

The input is an integer n , ($1 \leq n \leq \text{THE } n \text{ squares in the form of } Xn$, where X , ($A \times H$), is the column, and n , ($1 \leq n \leq 8$) is the row.

Note that there will be no black squares on A1 or A2.

Output

The output will be the number of steps for subtask A, followed by the number of steps for subtask B.

Example 1

Input

3
A6
E2
F5

Output

13
32

Example 2

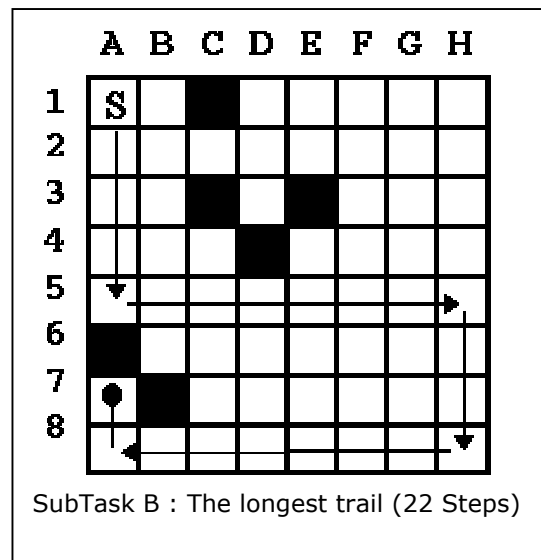
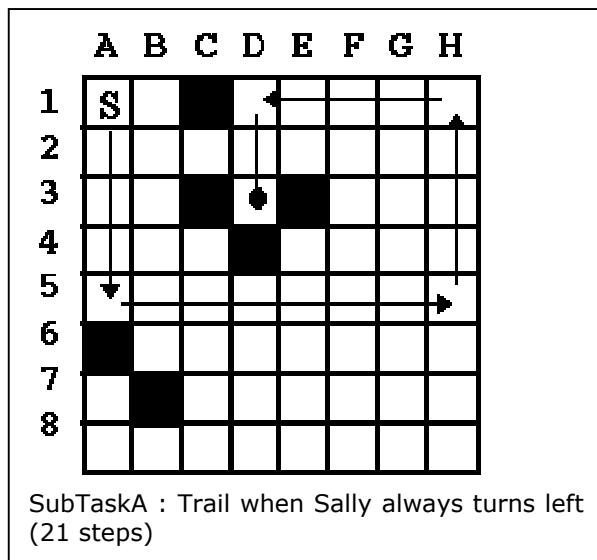
Input

6
A6
C1
C3
D4
E3
B7

Output

21
22

Here are the diagrams for example 2.



Day 2

Problem 1. Rotating Words

You are required to rotate a word a certain amount. For example, to rotate the word "Computer" by 1 results in "rCompute". Rotating it two more times gives you "terCompu".

Input

The first line contains the word (which will not have more than 15 letters). The second line contains an integer n, which will be less than the length of the word. You must rotate the word n times.

Output

The output will be the rotated word.

Example 1

Input

Computer
3

Output

terCompu

Example 2

Input

Program
1

Output

mProgra

Problem 2. Partial Problems

An integer substring of an integer is formed by consecutive digits of the original integer. For example, the number 6158 contains the substrings 6, 1, 5, 8, 61, 15, 58, 615, 158, and 6158. You must find the largest substring of an integer that is also a prime.

Input

The input will be an integer N, ($0 \leq N$)

Output

The output will be the largest prime substring of N. If no substring is a prime, then your program should print "No Primes"

Example 1

Input

2319

Output

31

Example 2

Input

6804

Output

No Primes

Problem 3. Letter Selection

Consider all possible selections (combinations) of the letters A, B and C :

A, AB, ABC, AC, B, BC, and C. Here they are sorted into dictionary order. If each selection is assigned an index then the selections correspond to the following indices.

1	A	There are two parts to the problem: Firstly you must find the index for a particular selection of letters. For example the index of the selection AC is 4. Secondly, you must find a selection n units below this. For example if n = 2, then the required selection is BC (two below AC).
2	AB	
3	ABC	
4	AC	
5	B	
6	BC	
7	C	

Input

The input will be three lines. The first line will be a number, k, ($1 \leq k \leq \text{THE LETTERS}$ k letters of the alphabet. (E.g. k = 5, then selections will be made of the five letters, A, B, C, D, and E.) The next line contains a particular selection. You will have to find its index.

The last line contains a number, n, ($0 \leq n \leq n$ to the index just found and find the corresponding selection.

Output

The output will be one line containing the index of the selection, and another line containing a selection corresponding to the modified index.

Example 1

Input

3
AC
2

Output

4
BC

(4 is the index of AC, BC is the selection 2 below it.)

Example 2

Input

4
AC
3

Output

6
B

Problem 4. Cube Chaos (Extra Problem)

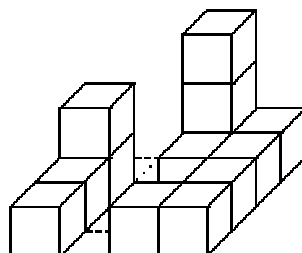
This problem will only be considered to distinguish between contestants whose scores are otherwise equal. You should ensure that you've got as many marks as possible in the other problems before attempting this one.

None of the test data will be seen, and there are no marks for producing an output file.

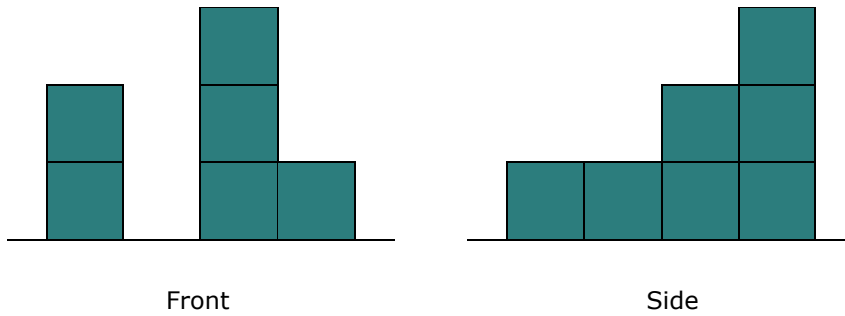
Otherwise this problem is the same as the others; you must save your program as 'day2d.ext' where ext is the extension (PAS, C, CPP or BC, BAS). Your program should read its input from a file called day2d.dat and write the solution to a file called day2d.sol.

Description

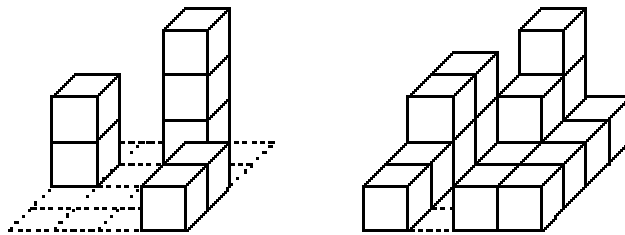
A construction is a number of cubes arranged on a square base as in the diagram below.



This is a four by four square supporting stacks of cubes. No stack will contain more than 8 cubes. The front and side projections of the construction are shown below.



Obviously there are a number of constructions that will satisfy both these projections. The constructions with the maximum and minimum number of blocks that will satisfy both projections are shown below.



Here the maximum number of blocks is 17, and the minimum is 7. Your job is to find the maximum and minimum number of blocks given the front and side projections.

Input

The first line contains k , ($1 \leq k \leq \text{THE LENGTH DESCRIPTION}$) k integers. Each integer indicates the height of the corresponding projection of a stack of blocks.

Output

The output will be one line containing the maximum number of blocks followed by a line containing the minimum number of blocks.

Example 1

Input

```
4
2 0 3 1
1 1 2 3
```

Output

```
17
7
```

Example 2

Input

```
1
1
1
```

Output

```
1
1
```


HS Computing Problems Archive

Polynomial Solver -- Difficulty **1/2 on up

Write a program to find (real) roots of polynomials. Read the polynomial's degree and then the coefficients (first number is the constant, then comes coefficient for x , then coefficient for $x*x$, etc.). Here's the input data for a polynomial whose solution is the positive and negative square roots of 2:

Degree: 2
Coefficients: -2 0 1

Which means: find the roots for a quadratic:

$$-2 + 0*x + 1*x*x$$

which is the same as

$$1*x*x + 0*x + -2$$

Which means: find the square roots of 2.

Keep track of how many times you evaluate the function and print that along with the solution:

Solution: $x=1.41421...$ (893 evaluations)
Solution: $x=-1.41421...$ (987 evaluations)

Be sure to check your answers (by plugging x back into the equation).

Try to minimize the number of evaluations to find the solution(s). I think an x is close enough to a root if the associated y value is within 0.000001 ($1.0e-6$) of 0. Just as a datapoint, my little tiny test program requires only four evaluations to find the first (positive) square root of two (i.e., for the sample input data above).

Test your program for, say, fifth degree polynomials that have more difficult roots to find.

Print a Generic Month -- Difficulty *1/2

Write a program that prints a formatted calendar month like this:

```
in--> First day, number days: 2,31

out--> S  M Tu  W Th  F  S
        1  2  3  4  5
        6  7  8  9 10 11 12
       13 14 15 16 17 18 19
       20 21 22 23 24 25 26
       27 28 29 30 31
```

Note that the 'first day' is 0 for Sunday, 1 for Monday, and so on through 6 for Saturday.

Print a Specific Month -- Difficulty **1/2

For months in the standard calendar (after the year of 1752), write a program that prints the actual calendar for a given month/year:

```
in--> Month, year: 8,1953

out-->      August 1953
        S  M Tu  W Th  F  S
           1
        2  3  4  5  6  7  8
       9 10 11 12 13 14 15
      16 17 18 19 20 21 22
      23 24 25 26 27 28 29
      30 31
```

Shuttle Puzzle [Piele] -- Difficult ***

The SHUTTLE PUZZLE of size 3 consists of three white marbles, three black marbles, and a strip of wood with 7 holes. The marbles of the same color are placed in the holes at the opposite ends of the strip leaving the center hole empty.

```

WWW BBB    <---  INITIAL STATE
BBB WWW    <---  GOAL STATE

```

There are only two types of moves you can use. You may move 1 marble 1 space (into the empty hole) or jump 1 marble over 1 marble of the OPPOSITE color (into the empty hole). You may not back up and you may not jump over 2 marbles.

A Shuttle Puzzle of size N consists of N white marbles and N black marbles and 2N+1 holes.

Write a program that will solve the SHUTTLE PUZZLE for any size $N \leq 10$ and display the board after each move. Use W's to represent white marbles and B's to represent black marbles and a blank to represent the empty hole. Test your program for $N=4$ and $N=5$.

```

Sample Run
N = 3
WWW BBB
WWWB BB
WW BWBB
W BWBBB
WBW WBB
WBWBW B
WBWBWB
WBWB BW
WB BWBW
  BWBWBW
B WBWBW
BBW BWB
BBWBW W
BBWB WW
BB BWWW
BBB WWW

```

Abundant Numbers [Sidney Kravitz] -- Difficulty **1/2 of 5

Positive integers can be classified as PERFECT, ABUNDANT, or DEFICIENT depending on the sum of their integer divisors. While calculating the sum of all divisors of an integer that are smaller than that integer, consider the following table:

N	Divisors	Sum	Classification
2	1	1	DEFICIENT (prime)
3	1	1	DEFICIENT (prime)
4	1,2	3	DEFICIENT
5	1	1	DEFICIENT (prime)
6	1,2,3	6	*PERFECT*
7	1	1	DEFICIENT (prime)
8	1,2,4	7	DEFICIENT
9	1,3	4	DEFICIENT
10	1,2,5	8	DEFICIENT
11	1	1	DEFICIENT (prime)
12	1,2,3,4,6	16	ABUNDANT

Numbers `n' for which the sum of the divisors is smaller than n are `deficient' (in fact, if the sum is 1 then the number n is prime).

Numbers `n' for which the sum of divisors is exactly equal to n are called `perfect'.

Numbers `n' for which the sum of divisors is greater than n are termed `abundant'.

Finding perfect numbers is keen. Finding amicable chains (in which the sum of divisors for n_1 is n_2 and the sum of divisors of n_2 is n_1 is even more fun (and there are longer chains than length 2).

Our question today is: Can you find consecutive ABUNDANT numbers? How many consecutive pairs are there between 1 and 10,000?

Feeding Game [Don Holstein Piele] -- * out of 5***

Humans have no idea what we cows think about. You see us resting peacefully under a shady tree, contented, unconcerned with the worries of the world, chewing our cud, and assume that nothing is going on between our ears.

But you're sadly mistaken. We think a lot about problems that interest us -- like our next meal. Let me say up front, I love my chow. Try producing 60 pounds of milk a day (on average) on an empty stomach or two. Being a cow who always likes to get its fair share, I've been doing some cowculations along this line.

My boss, farmer Paul, has been playing a weird game lately. When we come in for milking he's been setting out a row of eight pails, each filled with varying amounts of feed. (It looks to us as though they are randomly filled.) Two cows are assigned a row of pails and given the following instructions:

Each bucket is labeled with the number of scoops of feed inside. Going in turn, pick a pail from either end of the row, eat the contents, and discard the pail from the row. Take turns (waiting for the other to eat and remove the pail) until all the food is gone.

The winner of the game is, of course, the cow that gets the most feed.

Some of my casual bovine mates, taking a cavalier approach to this task, just flip a coin to decide which side to pick. They are easy to beat -- I just pick the end with the highest numbered pail and I get more food than they do. (I call this my Greedy Algorithm.) This usually works, but every once in a while they get lucky and beat me.

For example, the other day farmer Paul placed the following row of feed pails [1 2 3 2 4 2 15 6] for Bessie and me to eat. I went first and took bucket 6 (the right hand side) leaving [1 2 3 2 4 2 15]. Bessie picked 15 so it was clear I wasn't going to overcome her lead and I went a bit hungry that day.

Hunger may bring out the best in us, but I want to make sure it doesn't happen again. I want to be certain that I get an equal share or better -- no matter what! In other words, when farmer Paul places an even number of pails and fills each with a random amount of feed and I pick first (age has its advantages), I never want to end up with less than half the chow.

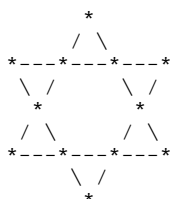
So here's your "Challenge Outta Wisconsin" or COW as we say around the dairy state: With your cowmputer, write a program that will fill p (p even ≤ 100) pails with random amounts of feed (integers between 1 and 99) and, by picking first, always win against any cow -- or human (they think they're so smart).

C-O-W A: Find an efficient algorithm I can use to win at chow time over Bessie -- every single time. Your solution should run within a few seconds, winning quickly even with 100 pails. Remember, I have the advantage of going first.

C-O-W B: Do some cowculations to estimate my chances of beating Bessie if we both use the Greedy algorithm. (Ties are considered a win for me.)

Product Star [Jaime & Juan Poniachik] -- * of 5**

Consider a star whose 12 points are the integers from 1..12:



No two points share the same value.

Find a way to place the integers 1..12 onto the points of the star such that the product of each of the 6 sets of four collinear integers is 1 mod 13.

Circular Sequences [Mihai Badoiu & others] -- Difficulty ** of 5**

A integer n is given. You have to compute a maximum circular sequence of numbers, using only 0 and 1.

A maximum circular sequence is one in which a consecutive sequence of length n must appear only once -- especially when the sequence is followed by itself.

Example solutions:

```
n=1    01
n=2    0011
n=3    00010111
```

Explanation:

```

Note that for n = 3, we have the following set of sequences:
.000.10111 first three bits of n=3
0.001.0111 second three bits of n=3
00.010.111 etc...
000.101.11
0001.011.1
00010.111.
000101.110.0010111 [This is where the `circular' part comes in]
0001011.100.010111
and we're back where we started...

```

It's easy to conjecture that $n=4$ is of length 16 and, generally, that lengths are 2^{*n} .

Superprime Beef [Piele, et al.] -- Difficulty: ***/5

Any farmer trying to make a living milking cows has to pay attention to good breeding. Just as race horses are bred for speed, cows can be bred for milk production. By "grading up the herd," as we say in Wisconsin, the average yearly production of milk has climbed to the current level of 12 to 15 thousand pounds per cow. Bessie, a purebred Holstein and barnyard companion, produced a record 44,000 pounds of milk one year; now that's a lot of butter.

Superprimes

Farmer Paul, my boss, and a member of the Primeville Organization for Outstanding Purebreds, takes great pride in raising some of the best purebred Holstein cows in Wisconsin. He affectionately calls them his superprime beef. To distinguish them from the rest of the herd, he brands them with a very special prime number -- a superprime. A superprime is any prime number that remains a prime when any number of digits are removed from the right side of the number. For example 5939333 is a superprime because 5939333 and 593933 and 59393 and 5939 and 593 and 59 and 5 are all prime.

Farmer Paul has been in the business for some time now and has seen his herd of superprime beef grow considerably. Predictably, he is running into difficulty coming up with new superprimes and has asked me for some assistance.

This month, between holiday shopping and all the barnyard parties, I've been ruminating about this problem. Of course, it has been known since the time of Euclid that there are an infinite number of primes. And the famous Prime Number Theorem asserts that the density of prime numbers, as measured by counting the number of prime numbers $\leq n$ grows as $n/\text{Log}[n]$.

For example, there are 78,498 prime numbers between 1 and 1,000,000.

So we are confident there are plenty of prime numbers to brand a herd of any size. But farmer Paul wants to use only superprimes. Will there always be enough numbers? What does your intuition tell you? OK, now go to your cowculator and find some, maybe all, superprimes.

Note: the number 1, by itself, is not a prime.

This problem was used as a problem at the training camp of the 1995 USA Computing Olympiad final round.

Repusprimes [Piele]

Superprimes have a built-in right-handed bias. There are left-handed farmers, too, who prefer to knock off their digits from the left side. If they remain prime down to the last digit, they are called repusprimes. (Read repus backwards to understand why.) Let's check to see if 5939333 is a repusprime.

```
5939333, 939333, 39333, 9333, 333, 33, 3
```

Only the first and last number are prime. But don't despair! Look at this:

```
739397, 39397, 9397, 397, 97, 7
```

They're all prime!

What can you say about the number of repusprimes? Will there always be enough repusprimes for the growing herds of purebred Holsteins belonging to our left-handed friends in the business?

OK cowhands, it's time to fire up your iron and produce the special primes we need-- and pronto! With all the chores to be done around here, farmer Paul has no time to waste on pokey algorithms. Speed counts!

Milknim [Jude Turian, traditional] -- Difficulty: ***/5

Farmer John likes to play a game with his neighbor, Farmer Bob. The store that buys their products can only stock N jugs of milk ($1 \leq N \leq 1000$). According to their game, on each turn they each must sell the store at most M jugs ($1 \leq M \leq 1000$). They alternate turns. During each turn, they sell as many jugs to the store as they want (though they must sell at least 1 jug and at most M jugs). Whoever goes last (sells to the store last, rendering it fully stocked) is the loser of the game.

It turns out that for a given N and M, one player can force a win. Farmer Bob always lets Farmer John choose whether he goes first or second, because Farmer John is not very good at the game. Farmer John would like you to write a program for his supercomputer that will tell him which turn to choose (1st or 2nd) so that he can force a win.

input.txt contains N and M, respectively on one line:
20 6

Given N and M, your program should print 1 if the first person to move can force a win or 2 if the second person to move can force a win:
2

The above test case is correct.

[Be sure to justify your answers when you submit your programs to the list.]

Calculator Language [South Pacific Contest, 1992] -- Difficulty: ***/5

Calculator Language (CL) supports assignment, positive and negative integers and simple arithmetic. The allowable characters in a CL statement are thus:

A..Z	variable names
0..9	digits
+	addition operator
-	subtraction operator
*	multiplication operator
/	integer division operator
=	assignment operator
()	brackets
_	negative sign (note that this is underscore, not minus)

All operators have the same precedence and are right associative, thus:

$15 - 8 - 3 = 15 - (8 - 3) = 10.$

As one would expect, brackets will force the expression within them to be evaluated first. Brackets may be nested arbitrarily deeply. An expression never has two operators next to each other (even if separated by a bracket), an assignment operator is always immediately preceded by a variable and the leftmost operator on a line is always an assignment. For readability, spaces may be freely inserted into an expression, except between a negative sign and a number. A negative sign will not appear before a variable. All variables are initialised to zero (0) and retain their values until changed explicitly.

Write a program that will accept and evaluate expressions written in this language. Each expression occupies one line and contains at least one assignment operator, and maybe more.

Input will consist of a series of lines, each line containing a correct CL expression. No line will be longer than 100 characters. The file will be terminated by a line consisting of a single #.

Output will consist of a series of lines, one for each line of the input. Each line will consist of a list of the final values of all variables whose value changes as a result of the evaluation of that expression. If more than one variable changes value, they should be listed in alphabetical order, separated by commas. If a variable changes value more than once in an expression, only the final value is output. A variable is said to change value if its value after the expression has been evaluated is different from its value before the expression was evaluated. If no variables change value, then print the message `No Change'. Follow the format shown below exactly.

Sample input

```
A = B = 4
C = (D = 2) * _2
C = D = 2 * _2
F = C - D
E = D * _10
Z = 10 / 3
```

Sample output

```

A = 4, B = 4
C = -4, D = 2
D = -4
No Change
E = 40
Z = 3

```

Factors and Factorials [NZ Contest, 1993, Division I] -- Difficulty: **/5

The factorial of a number N (written N!) is defined as the product of all the integers from 1 to N. It is often defined recursively as follows:

$$1! = 1$$

$$N! = N * (N-1)!$$

Factorials grow very rapidly -- $5! = 120$, $10! = 3,628,800$. One way of specifying such large numbers is by specifying the number of times each prime number occurs in it, thus 825 could be specified as (0 1 2 0 1) meaning no twos, 1 three, 2 fives, no sevens and 1 eleven.

Write a program that will read in a number N ($2 \leq N \leq 100$) and write out its factorial in terms of the numbers of the primes it contains.

Output will consist of a series of blocks of lines, one block for each line of the input (see below for example). Output is right justified in fields of width 3 and each line (except the last of a block, which may be shorter) should contain fifteen numbers. Any lines after the first should be indented.

See if you can match this output for 5! and 53!:

```

5! = 3 1 1
53! = 49 23 12 8 4 4 3 2 2 1 1 1 1 1 1

```

Fractional HP Calculators [Kolstad, 1997] Difficulty: */5**

HP has a new calculator that still uses the old stack methodology but keeps its numbers as fractions instead of as floating point numbers (since answers can then be calculated exactly instead of stored as slightly inaccurate decimal expansions like 0.333333333 for one third). Write a program to simulate this calculator. It is understood that certain larger numbers (those that won't fit into 32 bit long integers) will be unrepresentable. However, your program should ensure that no accuracy is lost when it is possible to avoid such loss by clever programming tricks.

Input to this problem will be found in file INPUT.TXT and will comprise lines that have either a fraction, resembling these fractions:

```

0/1
5/6
9/4

```

or an operator, which is one of: + - / * x and p . x means 'exit the program now' and p means 'print the top of the stack'. Input fractions will always be pairs of positive numbers separated by a slash.

Push fractions onto the stack as they are encountered. The stack will never be more than 100 fractions deep. Interpret the operators as operating on the top of the stack; here are some examples:

```

1/1 1/1 + p --> 2/1
3/1 1/1 - p --> 2/1
2/1 3/4 * p --> 3/2
5/6 1/2 / p --> 5/3

```

Be sure that you always print answers in their proper, reduced form.

Here is some sample input:

```

1/1
2/1
+
5/7
*
p
1/2
/
p
x

```

and here is the corresponding output:

```

15/7
30/7

```

Russ Cox's problem

here's a problem for anyone who feels like it. it is somewhat related to the usaco95 final round problem with the cow sequences.

let's say you have a length L sequence whose elements can range from 0 to $N-1$. your task is to find the smallest sequence that contains all possible length L sequences, given L and N .

were L 1 and N 2, you have possibilities of 0 and 1. the answer thus could be '01'.

were L 2 and N 1, you have 00, 01, 10, and 11. the smallest sequence that contains everything is '00110'.

generalize for any L or N . some interesting ones might be

1,1
2,2
3,3
4,4
5,5

russ

USACO 96 Fall Championship

Tue-Thu, December 3-5, 1996

The USA Computer Olympiad Fall Championship is a computer programming contest open to all pre-college students on the hs-computing@delos.com mailing list. Four problems are presented for solution.

Πρόβλημα 1. The Errant Physicist [New Zealand Contest, 1989]

The well-known physicist Alfred E Neuman is working on problems that involve multiplying polynomials of x and y . For example, he may need to calculate

$$(-x^8 y + 9x^3 - 1 + y) * (x^5 y + 1 + x^3)$$

yielding the answer

$$-x^{13}y^2 - x^{11}y + 8x^8y + 9x^6 - x^5y + x^5y^2 + 8x^3 + x^3y - 1 + y$$

Unfortunately, such problems are so trivial that the great man's mind keeps drifting off the job, and he gets the wrong answers. As a consequence, several nuclear warheads that he has designed have detonated prematurely, wiping out five major cities and a couple of rain forests.

Write a program to perform such multiplications and save the world.

The file of input data will contain pairs of lines, none longer than 80 characters. Stop your program when no more input is available. Each input line contains a polynomial written without spaces and without any explicit exponentiation operator. Exponents are positive non-zero unsigned integers. Coefficients are also integers, but may be negative. Both exponents and coefficients are less than or equal to 100 in magnitude. Each term contains at most one factor in x and one in y (i.e., at most one term of x to a power and likewise for y). For example, an input file for the above problem might be:

```
-x8y+9x3-1+y
x5y+1+x3
#
```

Your program must multiply each pair of polynomials in the input and print each product on a pair of lines, the first line containing all the exponents, suitably positioned with respect to the rest of the information, which is in the line below. See the representation above.

The following rules control the output format:

- Terms in the output line must be sorted in decreasing order of powers of x and, for a given power of x , in increasing order of powers of y .
- Factors of similar terms must be combined into a single term. For example,

$$42x^2y^3 - 40x^2y^3 \quad \text{should be shown as } 2x^2y^3$$

- Terms with a zero coefficient must not be displayed.
- Coefficients equal to 1 are not to be printed, except for the case of a constant term equal to 1.
- When the exponent is 1 (e.g., x to the first power), do not print the exponent
- Binary pluses and minuses (that is the pluses and minuses connecting terms in the output) have a single blank column both before and after.
- If the coefficient of the first output term is negative, it is preceded by a unary minus in the first column, with no intervening blank column. Otherwise, the coefficient itself begins in the first output column.
- The output can be assumed to fit into a single line of at most 80 characters in length.
- There are no blank lines printed between each pair of output lines.
- There are no blank space on the end of input lines
- There should be no blank spaces on the end of output lines.

The above example conforms to all those requirements.

Input file: INPUT.DAT

```
-x8y+9x3-1+y
x5y+1+x3
```

Output to screen: the product of the polynomials, as described above

Example output:

$$-x^{13}y^2 - x^{11}y + 8x^8y + 9x^6 - x^5y + x^5y^2 + 8x^3 + x^2y - 1 + y$$

- Problem 1 clarification:

a--3z+2b means (a) - (-3z) + 2b
a++3z+2b means (a) + (+3z) + 2b

Self Test Data #1:

```
-x8y+9x3-1+y
x5y+1+x3
x+y
x+y
45x4y3+-3x6
7y1++3y8-x+7x2
x+3x4--65x5
7y4-78y6+x3y3
x67y34+y67
x34+37y87x3+7
```

Πρόβλημα 2. Hamming Codes [Traditional, Kolstad]

Given N, B, and D: Find a set of N codewords ($1 \leq N \leq 64$), each of length B bits ($1 \leq B \leq 8$), such that each of the codewords is at least Hamming distance of D ($1 \leq D \leq 7$) away from each of the other codewords. The Hamming distance between a pair of codewords is the number of binary bits that differ in their binary notation. Consider the two codewords 0x554 and 0x234 and their differences (0x554 means the hexadecimal number with hex digits 5, 5, and 4):

```
0x554 = 0101 0101 0100
0x234 = 0010 0011 0100
```

Bit differences: xxx xx

Since five bits were different, the Hamming distance is 5.

Input file: INPUT.DAT

Format: N, B, D on a single line

Example input file:

```
16 7 3
```

Output to screen: N codewords, sorted, in decimal, ten per line. Example (but not only possible example):

```
0 7 25 30 42 45 51 52 75 76
82 85 97 102 120 127
```

Self Test Data #1:

```
16 7 3
```

Self Test Data #2:

```
8 6 3
```

Self Test Data #3:

```
8 5 2
```

Self Test Data #4:

```
16 8 3
```

Πρόβλημα 3. Palindromic Squares [Kolstad]

Palindromes are numbers that read the same forwards as backwards. The number 12321 is a typical palindrome.

Given a number base B ($2 \leq B \leq 20$ base 10) that has been read from the file INPUT.DAT, print all the integers N ($1 \leq N \leq 100$ base 10) such that the square of N is palindromic when expressed in base B; also print the value of that palindromic square. Use the letters 'A', 'B', and so on to represent the digits 10, 11, and so on.

Print both the number and its square in base B.

Input file: INPUT.DAT

Example input file:

```
10
```

Output to screen: the table of integers and their palindromic squares

Example output:

```
1 1
2 4
3 9
```

```
11 121
22 484
26 676
```

```
Self Test Data #1:
10
Self Test Data #2:
7
Self Test Data #3:
2
Self Test Data #4:
17
Self Test Data #5:
11
```

Πρόβλημα 4. Window Area [IV Balkan Olympiad]

You've just be assigned the project of implemented a windowing interface. This windowing interface is fairly simple, and fortunately, you don't have to display the actual windows. There are 5 basic operations:

- 1) Create a window
- 2) Bring a window to the top
- 3) Put a window to the bottom
- 4) Destroy a window
- 5) Output what percentage of a window is visible (i.e., isn't covered by windows above it).

In the input, the operations appear in the following format:

Create window:	w(I,x,y,X,Y)
Bring window to top:	t(I)
Put window on bottom:	b(I)
Destroy window:	d(I)
Output percentage visible:	s(I)

The I is a unique identifier for each window, which is one character. The character can be any of 'a'..'z', 'A'..'Z', and '0'..'9'. No extra spaces will appear in the input.

(x,y) and (X,Y) are opposite corners of the window. When a window is created, it is put 'on top'. You can't create a window with an identifier that is already in use, but you can destroy a window and then create a new one with the identifier of the destroyed window. Coordinates will be positive integers, and all windows will be of non-zero area ($x \neq X$ and $y \neq Y$).

Partial credit will be given for solving the following problems:

- 1) $0 < x,y < 250$
- 2) Able to solve up to 20 windows

However, to receive full credit, the program must work for any number of windows (limited by the identifiers, of course), and x and y coordinates between 1 and 32767 inclusive.

Input file: INPUT.DAT

The input file will consist of a sequence of commands to your interpreter. They will be listed one per line. Terminate the program when no more input is available

```
Input sample:
w(a,10,132,20,12)
w(b,8,76,124,15)
s(a)
```

Output to screen:

Output lines only for the s() commands. Of course, there might be several s() commands so the output should be a sequence of percentages, one per line, stating the percentage of the windows that are visible. The percentages should be rounded to 3 decimal place.

```
Sample Output:
49.167%
```

```
Self Test Data #1:
Input:
w(a,10,132,20,12)
w(b,8,76,124,15)
```

s(a)

```
Self Test Data #2:
Input:
```

```
w(a,1,1,21,21)
w(b,2,2,12,12)
w(c,5,5,8,8)
s(a)
s(c)
s(b)
```

Self Test Data #3:

```
Input:
w(a,164,16,78,102)
w(9,110,96,41,225)
w(d,60,89,141,158)
s(a)
b(d)
s(a)
t(a)
s(9)
s(d)
```

Self Test Data #4:

```
Input:
w(a,10,132,20,12)
w(c,12,120,22,16)
```

```
w(b,8,16,124,15)
t(a)
w(d,18,93,102,20)
b(b)
b(a)
s(a)
s(b)
s(c)
s(d)
d(d)
d(c)
s(a)
s(b)
```

Self Test Data #5:

```
Input:
w(a,4,452,182,338)
w(b,435,340,56,467)
w(c,474,4,343,486)
w(d,243,204,188,118)
w(e,493,173,301,68)
w(f,96,306,57,433)
w(g,362,344,313,105)
w(h,432,277,319,94)
s(a)
s(b)
s(c)
s(d)
```

USACO 96 Winter Championship

Πρόβλημα 1. Milk Containers [Piele, 1997]

Farmer Paul has many milk containers of each of the following sizes:

Can	10 gallons
Pail	2 gallons
Gallon	
Quart	1/4 gallon
Pint	1/8 gallon
Cup	1/16 gallon

Write a program that will compute the number of ways farmer Paul can store X gallons of milk using any combination of these containers. For instance, farmer Paul can store one Quart four ways:

- 1: 1 quart
- 2: 2 pints
- 3: 1 pint + 2 cups
- 4: 4 cups

One gallon can be stored 26 different ways.

In all data, X is a positive integer number and $1 \leq X \leq 50$. Your program must compute the number of combinations for each separate input value in less than ten seconds (which means that your program might run as long as $10 \cdot n$ seconds for n input values). Your program should read values from the file INPUT.DAT one per line (and compute and print the number of combinations) until encountering a value of 0.

SAMPLE INPUT (file INPUT.DAT):

```
5
10
0
```

SAMPLE OUTPUT:

```
5 1308
10 12477
```

TEST DATA SET #1:

```
1
5
10
15
39
0
```

Πρόβλημα 2. Super Roman Numerals [Kolstad, 1997]

You've heard the story of Romans like Midas who had the 'Golden Touch'. Midas was no fool and sold his gold for lots of money. The traditional Roman numerals proved inconvenient for expressing the value of his fortune, which ranged into the millions. He invented Super Roman Numerals.

Super Roman Numerals follow the traditional rules for Roman numerals but have many more single-character values. Consider the traditional Roman numeral values, shown here with the single letter and the decimal number it represents:

I	1	L	50	M	1000
V	5	C	100		
X	10	D	500		

As many as three of the same marks that represent 10^n may be placed consecutively:

```
III is 3
CCC is 300
```

Marks that are $5 * 10^n$ are never used consecutively.

Generally (with the exception of the next rule), marks are connected together and written in descending order:

$$\text{CCLXVIII} = 100+100+50+10+5+1+1+1 = 268$$

Sometimes, a mark that represents 10^n is placed before a mark of one of the two next higher values (I before V or X; X before L or C; etc.). In this case, the value of the smaller mark is SUBTRACTED from the mark it precedes:

$$\begin{aligned}\text{IV} &= 4 \\ \text{IX} &= 9 \\ \text{XL} &= 40\end{aligned}$$

But compound marks like XD, IC, and XM are not legal, since the smaller mark is too much smaller than the larger one. For XD (wrong for 490), one would use CDXC; for IC (wrong for 99), one would use XCIX; for XM (wrong for 990), one would use CMXC.

Regrettably, in standard Roman numerals, numbers like 10,000 are represented as MMMMMMMMMM. In Super Roman Numerals, the table of marks is extended:

I	1	L	50	M	1,000	R	50,000	U	1,000,000	N	50,000,000
V	5	C	100	P	5,000	S	100,000	B	5,000,000	Y	100,000,000
X	10	D	500	Q	10,000	T	500,000	W	10,000,000	Z	500,000,000

Numbers greater than 100 million are now easily expressible.

Write a program that reads non-negative decimal numbers (one per line) from the file INPUT.DAT and prints the Super Roman Numeral equivalent. It is promised that the input data will require an answer that is representable using the given rules. Stop your program when the input number is a 0.

SAMPLE INPUT (file INPUT.DAT):

18
1997
12345678
0

SAMPLE OUTPUT:

18 XVIII
1997 MCMXCVII
12345678 WUUSSSQRPDCLXXVIII

TEST DATA SET #1:

18
1998
87654321
99999999
11111
0

Πρόβλημα 3. Babylonian Fractions [Traditional]

In ancient Babylon, fractions were poorly understood. The strange way they represented fractions was as a sum of the minimal number of descending size terms of unique fractions with numerators of 1. Consider $2/3$:

$$2/3 = 1/2 + 1/6$$

Write a program that reads pairs of numbers, one per line, from the file INPUT.DAT (stopping when either of the numbers is 0). It should print the Babylonian equivalent of the quotient of that pair of numbers formatted similarly to the examples below.

Your program must compute the number of combinations for each separate input value in less than ten seconds (which means that your program might run as long as 10^n seconds for n input values).

- No solution will require a fraction whose denominator (bottom number) is greater than 10,000.
- The least common multiple (LCM) of any solution is promised to fit in a signed, 32 bit integer
- Some contestants found some **really** tough test cases. We will be revisiting the time constraints for this problem. Some tough cases can require up to 600 seconds of CPU. Don't worry if your solution takes that long for very tough cases.

SAMPLE INPUT (file INPUT.DAT):

```
2 3
3 4
1 2
0 0
```

SAMPLE OUTPUT (note the format):

```
2/3 = 1/2 + 1/6
3/4 = 1/2 + 1/4
1/2 = 1/2
```

TEST DATA SET #1:

```
2 3
3 4
1 2
79 1200
26 27
0 0
```

Πρόβλημα 4. Electrical Engineering Lab [Kolstad, 1997]

A poorly funded high school in Elbonia wishes to construct a laboratory for Electrical Engineering students. They are so poor that they wish to build a software laboratory. They are concerned only with resistors.

Resistors are electrical devices with two `terminals' (wires going into the resistor). Between these two terminals there is some amount of `resistance' measured in `ohms'. A single resistor is diagrammed (in the USA) somewhat like this:

```
----/\ /\ /\----
      10 ohms
```

There isn't a notion of an input or output terminal (wire) -- that all depends on which way electricity flows through the device. Furthermore, the device is perfectly symmetrical and bidirectional.

Hooking two resistors `in series' is diagrammed like this:

```
----/\ /\ /\-----/\ /\ /\----
      10 ohms       20 ohms
```

These two resistors in series behave precisely a single resistor whose value is the sum of the two resistors (10+20 in this example):

```
----/\ /\ /\----
      30 ohms
```

Another way (not the only other way) to connect resistors is known as connecting them `in parallel' and is diagrammed like this:

```
+----/\ /\ /\----+
----+   10 ohms   +-----
+----/\ /\ /\----+
      20 ohms
```

The resistors' left terminals is connected together as are the right terminals.

These two resistors in series behave precisely a single resistor whose value is calculated thusly (given two resistors whose values are R1 and R2 ohms):

$$R_{tot} = \frac{R1 * R2}{R1 + R2}$$

```
----/\ /\ /\----
      R      ohms
```

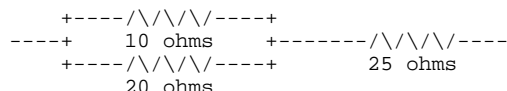
tot

For the example above (10 and 20 ohms), the equivalent value for a pair resistors of 10 and 20 ohms connected in parallel is:

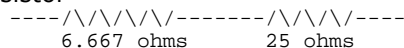
$$R_{tot} = \frac{R1 \cdot R2}{R1 + R2} = \frac{10 \cdot 20}{10 + 20} = \frac{200}{30} = 6.667 \text{ ohms}$$

Other configurations of resistors require more complex analysis and formulae.

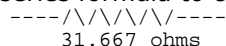
The Elbonians wish their software circuit simulator to be able to find equivalent resistances for sets of resistors connected in parallel and series (and combinations thereof, of course). For instance, they wish to calculate the equivalent resistance for a slightly more complex circuit that looks like this:



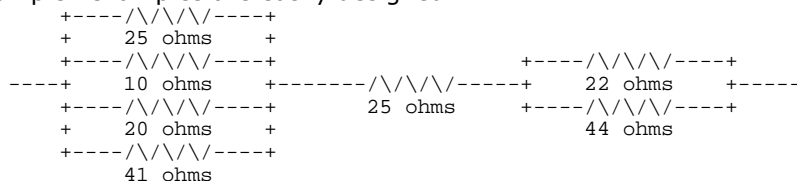
This circuit's equivalent resistance is found by finding the parallel resistance and substituting an equivalent transistor



then using the series formula to combine these two resistors:



More complex examples are easily designed:



This circuit is reduced pair-by-pair: maybe starting with the 25 and 10 ohm resistors in parallel, then the 41 and 20 in parallel, then those two equivalents in parallel, then the series combination of that equivalent with the 25 ohm resistor, and so on.

Write a program that accepts a set of resistor connections described by the resistors' values and endpoints:

```

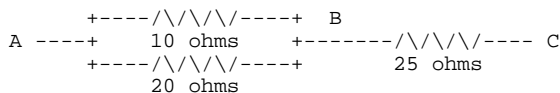
25 A B
100 B C
  
```

with a zero ohm resistor designating the endpoints (and designating the end of the input):

```

0 A C
  
```

Consider the previous example, now augmented with arbitrarily chosen SINGLE LETTER connection names:



whose input description is:

```

10 A B
20 A B
25 B C
0 A C
  
```

Connect names can be upper or lower case -- and node upper-case `A' is different from node lower-case `a'.

There will be no more than 500 resistors. Calculate your answers in floating point printing four decimal places of results. No formulae other than those presented here will be required (though

this notation enables potential test data that is not solvable with these equations -- such test data will not be presented to your program).

Read your input data from the file INPUT.DAT that has a single set of test data with the 0 ohm connection data as its last entry.

Print your answer with four decimal places:

31.6667

SAMPLE INPUT (file INPUT.DAT):

10 A B
20 A B
25 B C
0 A C

SAMPLE OUTPUT:

31.6667

TEST DATASET #1:

10 A B
20 A B
25 B C
0 A C

TEST DATASET #2:

10 A B
10 B C
10 C D
10 D E
10 E F
10 F G
10 G H
10 H I
10 I J
10 J K
10 K L
0 A L

TEST DATASET #3:

10 A B
10 B C
10 C D
10 D E
10 E F
10 F G
10 G H
10 H I
10 I J
10 J K
10 K L
10 A L
0 A L

TEST DATASET #4:

10 A B
10 A B
10 A B
10 A B
0 A B

TEST DATASET #5:

10 A B
10 B A
10 C D
10 D C
10 B C
10 A D
0 D A

Πρόβλημα 5. Planetary Timekeeping [Kolstad, 1997 after Heinlein]

Consider the Angusionian planetary system. It has N ($1 \leq N \leq 9$) planets circling the Angus sun in perfect circles. The system is so perfect that the planets trace circular, clockwise orbits that are all in the same plane.

Let's call the planets P_1, P_2, \dots, P_N and the respective radius of their orbits R_1, R_2, \dots, R_N . Any planet with orbit of radius 100 mooters requires exactly one year to orbit the sun. Planets closer to the sun require less time to complete an orbit; planets farther from the sun require more time. In fact, the length of a planet's year is precisely:

$$\text{Yearlen}_i = \left| \frac{R_i}{100} \right|^2 \text{ years}$$

So a planet whose orbit has radius 200 mooters would require $(200/100)^2 = 2^2 = 4$ years to complete an orbit. A planet whose orbit has radius 50 mooters would require $(1/2)^2 = 0.25$ years to complete an orbit.

You will be given a set of planets. For each planet, you will be given the size of its radius (in mooters) and its initial position for this observation. Its initial position will be expressed in clock time. The clock time represents the location of the planet in its orbit. At the specified time, the 'little hand' of an analog clock will point to the position of the planet in its orbit. By way of example, a planet located at 12:15 is 180 degrees around the sun from a planet positioned at 6:15.

Given the positions and orbital radii of a set of planets, you are to determine the date (in years) when the planets will all line up at 3:00 (i.e., all planets will be positioned in their orbits at the 3:00 angle).

Consider a single planet of orbital radius 100 mooters and located at 12:00. In 0.25 years, it will be at 3:00.

Consider a pair of planets. Planet 1 has radius 100 mooters and is currently located at 9:00. Planet 2 has radius 200 mooters and is located at 1:30. After 0.5 years, they will both be aligned at 3:00.

Your input file is named INPUT.DAT and includes a set of lines each with a radius and starting time. The last input line is all zeroes and is to be ignored. The input file looks like this for the previous example:

```
100 9 00
200 1 30
0 0 0
```

You are to calculate the number of years until the planets align at 3:00 and print the answer to four decimal places:

```
0.5000
```

It is guaranteed that the planets will eventually line up. The least common multiple of the planets' radii will be less than 2,000,000,000.

NOTE: 'Starting times' (initial angles) might have a floating point number for the minutes. A planet of radius 100 mooters might have an input line like this:

```
100 9 42.5
```

SAMPLE INPUT (file INPUT.DAT):

```
100 9 00
200 1 30
0 0 0
```

SAMPLE OUTPUT:

```
0.5000
```

TEST DATA SET #1:

```
100 9 00
200 1 30
0 0 0
```

TEST DATA SET #2:

```
100 9 00
200 7 30
0 0 0
```

TEST DATA SET #3:

```
100 9 00
200 1 30
300 10 20
0 0 0
```

TEST DATA SET #4:

```
100 12 00
200 8 15
300 9 20
75 8 20
0 0 0
```

TEST DATA SET #5:

```
5000 3 0
100 3 0
200 3 0
0 0 0
```

1997 USA Computing Olympiad

*National Championship Problems
Updated for International Competition
April 10, 1997*

Τα προβλήματα αυτά είναι στην ουσία μετατροπές προβλημάτων που έχουν ήδη χρησιμοποιηθεί σε διάφορους διαγωνισμούς.

Πρόβλημα 1. Γραμμάτωση

Given a set of N stamp values (e.g., {1 cent, 3 cents}) and an upper limit K to the number of stamps that can fit on an envelope, calculate the largest unbroken list of postages from 1 cent to M cents that can be created.

For example, consider stamps whose values are limited to 1 cent and 3 cents; you can use at most 5 stamps. It's easy to see how to assemble postage of 1 through 5 cents (just use that many 1 cent stamps), and successive values aren't much harder: $6 = 2*3$; $7 = 2*3+1$; $8 = 2*3+2*1$; $9 = 3*3$; $10 = 3*3+1$; $11 = 3*3+2*1$; $12 = 4*3$; $13 = 4*3+1$.

However, there is no way to make 14 cents of postage with 5 or fewer stamps of value 1 and 3 cents. Thus, for this set of two stamp values and a limit of $K=5$, the answer is $M=13$.

The first line of the input file has K , the total number of stamps that can be used, followed by N , the number of stamp values. The second and subsequent lines list all the N stamp values, 15 per line. Your job is to compute and print M , the number of contiguous postage values (start at one cent) that can be formed using no more than K stamps from the set.

It is promised that $1 \leq N \leq 500$ and $1 \leq K \leq 500$. No stamp value will exceed 10,000. Long integers (signed 32-bit) will be adequate for all solutions. It is not promised that it is possible to write a program that can solve a large case within the allotted time limit.

SAMPLE INPUT (file INPUT.TXT):

```
5 2
1 3
```

SAMPLE OUTPUT (file OUTPUT.TXT):

```
13
```

TEST DATA SET #1	TEST DATA SET #2	TEST DATA SET #3
5 2	6 3	8 4
1 3	2 4 7	1 2 5 9

Πρόβλημα 2. Runaround Numbers

Runaround numbers are integers with unique digits, none of which is zero (e.g., 81362). Furthermore, they have a neat property, exemplified by this demonstration:

- If you start at the left digit (8 in our number) and count that number of digits to the right (wrapping back to the left side when no digits on the right are available), you'll end up at a new digit (a number which does not end up at a new digit is not a Runaround Number). Consider: 8 1 3 6 2 which cycles through eight digits: 1 3 6 2 8 1 3 6 so the next digit is 6.
- Repeat this cycle (this time for six counts) and you should end on a new digit: 2 8 1 3 6 2, namely 2.
- Repeat again (two digits this time): 8 1
- Continue again (one digit this time): 3
- One more time: 6 2 8 and you have ended up back where you started, after touching each digit once. If you don't end up back where you started after touching each digit once, your number is not a Runaround number.

Given a number M , find and print the next runaround number higher than M .

SAMPLE INPUT (file INPUT.TXT):

```
81361
```

SAMPLE OUTPUT (file OUTPUT.TXT):
81362

TEST DATA SET #1	TEST DATA SET #2	TEST DATA SET #3
81360	123456	1234567

Πρόβλημα 3. Humble Numbers

For a given set of K prime numbers $S = \{p_1, p_2, \dots, p_K\}$, consider the set of all numbers whose prime factors are a subset of S . This set contains, for example, p_1 , p_1p_2 , p_1p_1 , and $p_1p_2p_3$ (among others and in no particular order).

This is the set of 'humble numbers' for the input set S . Note: The number 1 is explicitly declared not to be a humble number. Your job is to find the N th humble number for a given set S . [Consider the unique members of set S ordered from smallest to largest. Call them H_1, H_2, H_3 , and so on. The N th humble number is HN .]

The input file has K and N respectively on the first line with K primes on the next line.

It is promised that $1 \leq K \leq 100$ and $1 \leq N \leq 100,000$. Long integers (signed 32-bit) will be adequate for all solutions. It is not promised that it is possible to write a program that can solve a large case within the allotted time limit.

SAMPLE INPUT (file INPUT.TXT):

```
4 19
2 3 5 7
```

SAMPLE OUTPUT (file OUTPUT.TXT):

27

TEST DATA SET #1 TEST DATA SET #2 TEST DATA SET #3

4 19	5 50	2 75
2 3 5 7	2 3 11 13 17	2 3

Πρόβλημα 4. Digit Pals

Consider a game played with matrix of M rows and N columns of digits 0...9, for example:

		row\col	a	b	c	d	e
1	3	5	2	2			
2	2	3	5	1			
1	2	3	5	5			

which is referenced as row,col using indexes shown here:

3:	1	3	5	2	2
2:	2	2	3	5	1
1:	1	2	3	5	5

The lower left corner is 1a and the upper right corner is 3e.

This game involves removing sets of digits from the matrix according to the following rules:

1. Digit pals are digits of the same value which are 4-connected (adjacent either up/down or right/left; diagonal adjacency is not considered to be 4-connected). You pick a digit and remove it and all its pals and all their pals, etc. Isolated digits (those without pals) cannot be removed. In the example above, the 1's are currently all isolated and cannot be removed. You can remove two sets of digit pals that contain the digit 2 and one each digit pals containing 3 or 5.
2. When you remove digits from a column, all the digits slide down to stack nicely at the bottom of the column. See the examples below.
3. If there is an empty column, i.e., the bottom row of that column is blank, the column is deleted and all the columns to its right slide left (as many times as necessary) to fill in the gap. The matrix might have new neighbors and perhaps new pals or might be completely empty.
4. The object is to remove all the pals and be left with an empty matrix. If you are left with a non-empty matrix of isolated digits, you lose the game.

For example, if you are facing

```
1 2
2 1
```

then you lose the game.

Here is an example of a win using the 15 element matrix above.

First select 3e and remove the digit 2 pals:

```
1 3 5
2 2 3 5 1
1 2 3 5 5
```

Next, select 2b and remove the remaining digit 2 pals.

The 3 slides down:

```
5
1 3 5 1
1 3 3 5 5
```

Next, select 1b and remove the digit 3 pals. The 5 flows down and the right-hand three columns slide to the left.

```
1 5 1
1 5 5 5
```

Next select 1c (or 1b or 2c) to take out the digit 5 pals:

```
1
1 1
```

Finally, select 1a to empty the grid. You win! There were several different ways to show the sequence of plays to win this game.

Given sets of matrices, print an ordered sequence of regions to be removed that will win the game or print 'NO WIN'. If there are multiple sequences that win the game, you need print only one.

INPUT FORMAT

The first line of the input file is two integers representing the number of rows and number of columns. On each subsequent line, the numbers in the matrix are listed in column order starting at column a. The sample input shows the example discussed above.

The matrix will contain no more than 60 rows and no more than 26 columns. It is not promised that it is possible to write a program that can solve a large case within the allotted time limit.

SAMPLE INPUT (file INPUT.TXT):

```
3 5
1 3 5 2 2
2 2 3 5 1
1 2 3 5 5
```

SAMPLE OUTPUT (file OUTPUT.TXT):

3e 2b 1b 1c 1a

TEST DATA SET #1	TEST DATA SET #2	TEST DATA SET #3
3 5	4 5	3 5
1 3 5 2 2	1 3 5 2 2	1 3 5 2 2
2 2 3 5 1	2 2 3 5 1	2 2 3 5 1
1 2 3 5 5	1 2 3 5 5	5 1 2 3 5
	2 3 5 5 1	

USACO 97 Fall Championship

Δόθηκαν 5 ώρες για την επίλυση των 4 θεμάτων

Problem 1: SNAIL TRAIL [All Ireland Contest]

Στη Σάλυ το σαλιγκάρι, της αρέσει να κάνει βόλτα σε ένα $N \times N$ τετράγωνο πλέγμα ($1 < N < 120$). Πάντα ξεκινάει από την επάνω αριστερή γωνία. Το πλέγμα έχει άδεια τετράγωνα (σημειώνονται με '.') και έναν αριθμό (B) από εμπόδια (σημειώνονται με '#'). Παρακάτω είναι ένα παράδειγμα του πλέγματος το οποίο δείχνει ακριβώς τον τρόπο με τον οποίο σημειώνουμε στο πλέγμα:

	A	B	C	D	E	F	G	H
1	S	#	.
2	#	.	.	.
3
4
5	#	.	.
6	#
7
8

Η Σάλυ περπατάει πάντα κάθετα (πάνω ή κάτω) ή οριζόντια (αριστερά ή δεξιά). Μπορεί να περπατήσει ή προς τα κάτω ή προς τα δεξιά από την αρχική της θέση, η οποία είναι πάντα η A1.

Επίσης, η Σάλυ περπατάει όσο μπορεί πιο πολύ προς την κατεύθυνση που επέλεξε. Σταματάει και γυρνάει 90 μοίρες όποτε βρίσκει το τέλος του πλέγματος ή ένα από τα φράγματα. Δεν μπορεί να φύγει από το πλέγμα ή να μπει σε ένα τετράγωνο που έχει φράγμα. Ακόμα, η Σάλυ δεν μπορεί να πατήσει πάνω σε ένα τετράγωνο στο οποίο έχει ξαναπερπατήσει. Σταματάει όταν πια δεν μπορεί να κάνει άλλη κίνηση.

Εδώ είναι ένα παράδειγμα της βόλτας της Σάλυ στο παραπάνω πλέγμα:

	A	B	C	D	E	F	G	H
1	S	-----+					#	.
2	#		.	.
3
4	+---		
5	#	.	
6	#	
7	+	-----+						
8	+	-----+						

Η Σάλυ πήγε δεξιά, κάτω, δεξιά, κάτω, αριστερά, πάνω και δεξιά. Δεν μπόρεσε να συνεχίσει γιατί βρήκε μπροστά της ένα τετράγωνο το οποίο είχε ήδη επισκεφθεί. Τα πράγματα μπορούσαν να είχαν γίνει διαφορετικά, αν είχε διαλέξει να πάει αριστερά όταν συνάντησε το εμπόδιο στο F5.

Εσείς, πρέπει να βρείτε και να τυπώσετε το μέγιστο αριθμό από τετράγωνα τα οποία μπορεί να επισκεφθεί η Σάλυ, αν αποφασίζει σωστά. Μη ξεχάστε να μετρήσετε και το A1 σαν ένα τετράγωνο που επισκέφθηκε η Σάλυ.

INPUT FORMAT

Η πρώτη γραμμή του input έχει το N, το μέγεθος του τετράγωνου πλέγματος, και το B, τον αριθμό των εμποδίων ($1 \leq B \leq 200$). Στις επόμενες B γραμμές περιέχονται η τοποθεσίες των εμποδίων. Το παρακάτω παράδειγμα περιγράφει το παραπάνω πλέγμα. Το παράδειγμα του output file, περιγράφει τον δρόμο που ακολούθησε η Σάλυ στο παραπάνω παράδειγμα. Σημειώστε ότι όταν το $N > 26$ τότε το αρχείο input δεν μπορεί να προσδιορίσει εμπόδια δεξιότερα της στήλης Z.

OUTPUT FORMAT

Το output file θα περιέχει μόνο μία γραμμή, τον μεγαλύτερο αριθμό από τετράγωνα που μπορεί να επισκεφτεί η Σάλυ.

SAMPLE INPUT (file INPUT.TXT):

```
8 4
E2
A6
G1
F5
```

SAMPLE OUTPUT (file OUTPUT.TXT):
29

Πρόβλημα 2: Ο τετράγωνος Αχυρώνας

Ο Γιάννης ο αγρότης θέλει να φτιάξει έναν μεγάλο τετράγωνο αχυρώνα. Ο Γιάννης δεν θέλει να κόψει δέντρα στο χωράφι του και θέλει να βρεί ένα μέρος το οποίο να του δίνει τη δυνατότητα να φτιάξει τον αχυρώνα χωρίς να κόψει δέντρα. Για δικούς μας λόγους, το χωράφι θεωρούμε ότι χωρίζεται σε $N \times N$ κομμάτια ($1 \leq N \leq 200$).

Θα δοθεί μία λίστα των κομματιών που περιέχουν δέντρα. Η δουλειά σας είναι να βρείτε και να τυπώσετε το μέγιστο τετράγωνο στο οποίο μπορεί να φτιαχτεί ο αχυρώνας χωρίς να χρειαστεί να κοπούν δέντρα.

Παράδειγμα

Θεωρείστε το παρακάτω σχήμα του χωραφιού του Γιάννη, όπου '.' είναι ένα κομμάτι άδεια και '#' ένα κομμάτι με δέντρο:

```

  1 2 3 4 5 6 7 8
1 . . . . . . .
2 . # . . . # .
3 . . . . . . .
4 . . . . . . .
5 . . . . . . .
6 . . # . . . .
7 . . . . . . .
8 . . . . . . .
```

Ο μεγαλύτερος αχυρώνας που μπορεί να κατασκευαστεί είναι 5×5 και θα τοποθετηθεί σε μία από τις δύο τοποθεσίες που χωράει, κάτω δεξιά στο χωράφι.

INPUT FORMAT

Η πρώτη γραμμή περιέχει το N , το μήκος της μίας πλευράς του χωραφιού, και το T , τον αριθμό των κομματιών με δέντρα ($1 \leq T \leq 500$). Οι επόμενες γραμμές περιέχουν τις θέσεις των δέντρων ($1 \leq$ κάθε συντεταγμένη $\leq N$).

OUTPUT FORMAT

Το output file θα περιέχει μόνο μία γραμμή, που θα είναι η μέγιστη δυνατή πλευρά του αχυρώνα.

SAMPLE INPUT (file INPUT.TXT):

```
8 3
2 2
2 6
6 3
```

SAMPLE OUTPUT (file OUTPUT.TXT):

```
5
```

Πρόβλημα 3: Πρόλογοι με Λατινικούς Αριθμούς

Σε μερικά βιβλία, οι πρόλογοι αριθμούνται με κεφαλαίους Ρωμαϊκούς αριθμούς. Οι παραδοσιακοί Ρωμαϊκοί αριθμοί χρησιμοποιούν ένα γράμμα για να παραστήσουν έναν δεκαδικό αριθμό. Αυτό είναι το παραδοσιακό σετ:

I	1	L	50	M	1000
V	5	C	100		
X	10	D	500		

Το πολύ τρία όμοια σύμβολα που παριστάνουν το 10^n μπορούν να τοποθετηθούν συνεχόμενα για να φτιάξουν άλλους αριθμούς:

```
III = 3
CCC = 300
```

Τα σύμβολα που δηλώνουν το $5 \cdot 10^n$ δεν χρησιμοποιούνται ποτέ συνεχόμενα.

Γενικά (με μόνη εξαίρεση τον επόμενο κανόνα), τα σύμβολα ενώνονται μεταξύ τους και γράφονται σε κατιούσα σειρά για να φτιάξουν ακόμα περισσότερους αριθμούς:

```
CCLXVIII = 100+100+50+10+5+1+1+1 = 268
```

Μερικές φορές, ένα σύμβολο που έχει τιμή 10^n μπαίνει πριν από το σύμβολο της επόμενης ή της μεθεπόμενης τιμής (π.χ. το I πριν από V ή το X; το X πριν από το L ή το C; κ.λ.π.). Σ' αυτήν την περίπτωση, η τιμή του μικρότερου συμβόλου ΑΦΑΙΡΕΙΤΑΙ από την τιμή του συμβόλου που έπεται:

IV = 4
IX = 9
XL = 40

Συνδυασμοί συμβόλων όπως XD, IC, και XM δεν επιτρέπονται, γιατί το μικρό σύμβολο είναι πολύ μικρότερο του μεγάλου. Για το XD (που είναι λάθος για το 490), θα γράφαμε CDXC, για το IC (που είναι λάθος για το 99), θα γράφαμε XCIX, και για το XM (που είναι λάθος για το 990), θα γράφαμε CMXC.

Δοθέντος του N, του αριθμού των σελίδων του προλόγου ενός βιβλίου, βρείτε και τυπώστε τον αριθμό από I, V, κ.λ.π. (στη σειρά από το μικρότερο μέχρι το μεγαλύτερο) που απαιτούνται για να γραφτούν όλοι οι αριθμοί των σελίδων (σε Ρωμαϊκή γραφή) από το 1 μέχρι το N. Μην τυπώσετε γράμματα τα οποία δεν εμφανίζονται στους αριθμούς που ζητήθηκαν.

Παράδειγμα

Αν το N = 5, τότε οι αριθμοί των σελίδων θα είναι: I, II, III, IV, V. Ο συνολικός αριθμός από I είναι 7 και ο αριθμός των V είναι 2.

INPUT FORMAT

Το input file έχει μία μόνο γραμμή που έχει τον αριθμό P ($1 \leq P \leq 3500$) ο οποίος δηλώνει τον αριθμό των σελίδων του προλόγου.

OUTPUT FORMAT

Οι γραμμές του output δείχνουν, σε αύξουσα σειρά των Ρωμαϊκών αριθμητικών συμβόλων το γράμμα, ένα μόνο space, και το πλήθος των εμφανίσεων του γράμματος στους αριθμούς των σελίδων.

SAMPLE INPUT (file INPUT.TXT):
5

SAMPLE OUTPUT (file OUTPUT.TXT):
I 7
V 2

Πρόβλημα 4: Το Βοδήλατο [Don Gillies; Adapted by Galperin, Burch, Kolstad, 1995]

[This problem uses a cow metaphor. International readers should note that some words are puns on cows.]

Εχοντας κάνει μία περιουσία στο περιοδικό "ΠλέχΒόδι", ο Χάφ Βοδινός μετακόμισε από το χωράφι του στην εξοχή, σε μία όμορφη αυλή στα περίχωρα. Για να θυμηθεί όμως τις όμορφες ποιμενικές αναμνήσεις του, θέλει να βοδηγήσει πίσω στα παλιά βοσκοτόπια. Ως γνήσιος οικολόγος όμως, ο Χαφ θέλει να μετακινηθεί χρησιμοποιώντας το βοδήλατό του (ένα ποδήλατο που είναι ειδικό για τις περιποιημένες του οπλές).

Ο Χαφ ζυγίζει πάνω από έναν τόνο. Ετσι, το να βοδηγήσει ομαλά στο παραδοσιακό πολυτάχυτο βοδήλατο, είναι λίγο δύσκολο. Οι αλλαγές ταχυτήτων με μεγάλες διαφορές αναλογιών, προκαλούν καρδιακά προβλήματα στον Χαφ.

Βοηθείστε τον Χαφ να φτιάξει το βοδήλατό του, επιλέγοντας F ($1 \leq F \leq 5$) γρανάζια για μπροστά και R ($1 \leq R \leq 10$) γρανάζια για πίσω στο F*R-τάχυτο βοδήλατο, βάσει των παρακάτω κανόνων:

Ο αριθμός των δοντιών για τα F μπροστινά γρανάζια δίνεται.

Ο αριθμός των δοντιών για τα R πίσω γρανάζια δίνεται.

Σε κάθε ταχύτητα, η αναλογία της είναι το πηλίκο του αριθμού των δοντιών του μπροστινού γραναζιού και του αριθμού των δοντιών του πίσω γραναζιού (δηλ. ο αριθμός των μπροστά δοντιών διά του αριθμού των πίσω δοντιών)

Για κάθε σύνολο από F μπροστινά γρανάζια και R πίσω γρανάζια, η περιοχή των αναλογιών των ταχυτήτων πρέπει να καλύπτει τουλάχιστον έναν συντελεστή 3x.

Η διακύμανση του συνόλου των ΔΙΑΦΟΡΩΝ μεταξύ συνεχόμενων αναλογιών ταχυτήτων πρέπει να ελαχιστοποιηθεί.

Υπολογίστε τον μέσο όρο και τη διακύμανση του συνόλου των διαφορών (το x έχει δείκτη i στους τύπους) από τους παρακάτω τύπους:

$$\text{mean} = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$
$$\text{variance} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Βγάλτε συμπέρασμα και τυπώστε το ιδανικό σετ από F μπροστινά γρανάζια και R πίσω γρανάζια, έτσι ώστε η διακύμανση να είναι ελάχιστη (και οι αναλογίες ταχυτήτων να καλύπτουν ένα συντελεστή της τάξης του $3x$ τουλάχιστον).

INPUT FORMAT

Η πρώτη γραμμή περιέχει το F και το R, τους αριθμούς των μπροστά και πίσω γραναζιών. Η δεύτερη γραμμή περιέχει τέσσερα νούμερα: F1, F2 ($25 \leq F1 < F2 \leq 80$), R1, και R2 ($5 \leq R1 < R2 \leq 40$). Όλα τα γρανάζια μεταξύ F1 και F2 διατίθενται και όλα τα γρανάζια μεταξύ R1 και R2.

OUTPUT FORMAT

Γράψτε τον αριθμό των δοντιών του συνόλου των F επιλεγμένων μπροστινών γραναζιών, από το μικρότερο στο μεγαλύτερο, στην πρώτη γραμμή του output (χωρίστε τα με κενά).

Γράψτε τον αριθμό των δοντιών του συνόλου των R επιλεγμένων πίσω γραναζιών, από το μικρότερο στο μεγαλύτερο, στη δεύτερη γραμμή του output. Όλα τα γρανάζια έχουν ακέραιο αριθμό δοντιών φυσικά.

TIME LIMIT

Το χρονικό όριο για μερικά από τα datasets μπορεί να επεκταθεί και μέχρι 10".

SAMPLE INPUT (file INPUT.TXT):

```
2 5
39 62 12 28
```

SAMPLE OUTPUT (file OUTPUT.TXT):

```
39 53
12 13 15 23 27
```

(Αυτή η απάντηση θεωρείται σωστή και πιστεύουμε ότι έχει διακύμανση 0.007848921; αν βρείτε καλύτερη παρακαλούμε επικοινωνήστε με τον kolstad@bsd.com.)

USACO 98 Winter Championship

Δόθηκαν 5 ώρες για την επίλυση των 4 θεμάτων

Πρόβλημα 1: Τι ώρα είναι; [Richard Forster -- British Olympiad]

Πριν από χρόνια, ο Brian Redman έκανε ολόκληρη την κοινότητα των δικτύων να γελάσουν, όταν έκανε την ερώτηση: "Τί ώρα είναι; Γράψτε μου E-mail και εγώ θα τα μαζέψω και θα το πω στο δίκτυο." Χα χα.

Γράψτε ένα πρόγραμμα που να μετατρέπει την ώρα που είναι εκφρασμένη σε ώρα:λεπτά στην (όμορφη, δείτε πιο κάτω) μορφή της Αμερικάνικης Αγγλικής Γλώσσας.

Αυτοί είναι οι κανόνες για να το κάνετε (προσέξτε ότι μπορεί να είναι διαφορετικοί απ' αυτούς που έχετε συνηθίσει και είναι σίγουρα διαφορετικοί από τους Βρετανικούς κανόνες):

- Κάντε κεφαλαίο το πρώτο γράμμα του output σας
- Σύνθετοι αγγλικοί αριθμοί θα έχουν παύλα, δηλαδή **forty-four**
- Εκφράστε το **x:00** σαν **[x_in_english] o'clock**
- Εκφράστε το **x:15** σαν **Quarter past [x_in_english]**
- Εκφράστε το **x:30** σαν **[x_in_english] thirty**
- Εκφράστε το **x:45** σαν **Quarter to [next_hour_in_english]**
- Αλλιώς, εκφράστε το **x:nn** σαν:
- **[x_in_english] [nn_in_english]** όταν $nn < 45$
- **[60-nn_in_english] to [next_hour_in_english]** όταν $nn > 45$

Παραδείγματα:

5:00	Five o'clock
10:10	Ten ten
9:22	Nine twenty-two
5:15	Quarter past five
2:30	Two thirty
6:40	Six forty
5:45	Quarter to six
8:47	Thirteen to nine

INPUT FORMAT

Το αρχείο input έχει μία μόνο γραμμή που περιέχει την ώρα εκφρασμένη σαν:

ώρα:λεπτά

Κάθε ώρα ανήκει στο σύνολο 1..12; τα λεπτά είναι πάντα εκφρασμένα σε δύο ψηφία και ανήκουν στο σύνολο 0..59

OUTPUT FORMAT

Το output αρχείο θα έχει μία μόνο γραμμή που θα έχει τη "μετεφρασμένη" έκδοση της ώρας.

SAMPLE INPUT (file INPUT.TXT):

10:11

SAMPLE OUTPUT (file OUTPUT.TXT):

Ten eleven

Πρόβλημα 2: DUAL PALINDROMES
[from Mario Cruz (Colombia), by Hugo Rickeboer (Argentina?)]

Ενας αριθμός που διαβάζεται το ίδιο από δεξιά προς τα αριστερά όπως και όταν διαβάζεται από αριστερά προς τα δεξιά, ονομάζεται παλινδρομικός. Ο αριθμός 12321 είναι παλινδρομικός. Ο αριθμός 77778 δεν είναι. Όπως είναι φυσικό, οι παλινδρομικοί δεν έχουν ούτε στην αρχή, ούτε στο τέλος μηδενικά, δηλαδή ο αριθμός 0220 δεν είναι παλινδρομικός.

Ο αριθμός 21 (με βάση 10 - στο δεκαδικό σύστημα δηλαδή) δεν είναι παλινδρομικός, όμως ο 21 (βάση 10) είναι παλινδρομικός όταν γραφτεί στο δυαδικό σύστημα, με βάση 2 δηλαδή (10101).

Γράψτε ένα πρόγραμμα που διαβάζει δύο αριθμούς (στο δεκαδικό σύστημα):

- N ($1 \leq N \leq 15$)
- S ($0 < S < 10000$)

και μετά βρίσκει και τυπώνει (στο δεκαδικό σύστημα) τους πρώτους N αριθμούς αυστηρά μεγαλύτερους του S οι οποίοι είναι παλινδρομικοί σε δύο ή περισσότερα αριθμητικά συστήματα με βάση 2 έως 10.
($2 \leq \text{βάση} \leq 10$).

Οι λύσεις γι' αυτό το πρόβλημα δεν χρειάζονται ακεραίους μεγαλύτερους από τους standard 32 bit.

INPUT FORMAT

Το αρχείο input περιέχει μία μόνο γραμμή με δύο (δεκαδικού συστήματος)ακεραίους: N and S.

OUTPUT FORMAT

Το αρχείο output πρέπει να έχει N γραμμές, κάθε μία με έναν δεκαδικό αριθμό ο οποίος είναι παλινδρομικός όταν εκφραστεί σε τουλάχιστον δύο από τα αριθμητικά συστήματα 2..10.

SAMPLE INPUT (file INPUT.TXT):

3 25

SAMPLE OUTPUT (file OUTPUT.TXT):

26
27
28

Πρόβλημα 3: Μοσχαρίσια McNUGGETS [Hubert Chen]

Οι αγελάδες του μπάρμπα Στάθη έχουν πάρει τα όπλα, όταν άκουσαν ότι τα McDonalds σκέφτονται να ξεκινήσουν ένα νέο προϊόν: Μοσχαρίσια McNuggets. Οι αγελάδες προσπαθούν να βρουν οποιονδήποτε δυνατό τρόπο για να δώσουν στο προϊόν μία αρνητική εικόνα.

Μία στρατηγική που προσπαθούν να επιδιώξουν είναι αυτή της "υποδεέστερης συσκευασίας". "Δείτε," λέν οι αγελάδες, "αν έχετε Μοσχαρίσια McNuggets σε κουτιά των 3, 6, και 10, δεν μπορείτε να ικανοποιήσετε έναν πελάτη που θέλει 1, 2, 4, 5, 7, 8, 11, 14, ή 17 McNuggets. Κακή συσκευασία: κακό προϊόν."

Βοηθήστε τις αγελάδες. Σας δίνουμε το N (τον αριθμό των πιθανών επιλογών συσκευασίας, $1 \leq N \leq 10$), και ένα σύνολο από N θετικούς αριθμούς ($1 \leq i \leq 255$) οι οποίοι δείχνουν τον αριθμό των nuggets στις διάφορες συσκευασίες. Βρείτε τον μεγαλύτερο αριθμό από nuggets που δεν μπορεί να αγοραστεί, παίρνοντας nuggets των συγκεκριμένων μεγεθών. Τυπώστε 0 αν όλες οι πιθανές αγορές μπορούν να γίνουν, ή αν δεν υπάρχει όριο για τον μεγαλύτερο αριθμό.

Ο μεγαλύτερος αδύνατος αριθμός (αν υπάρχει) δεν θα είναι μεγαλύτερος από 2,000,000,000.

INPUT FORMAT

Ο αριθμός N (που δείχνει πόσες συσκευασίες υπάρχουν) εμφανίζεται στην πρώτη γραμμή του αρχείου input. Σε κάθε μία από τις επόμενες N γραμμές υπάρχει ένας ακέραιος που δείχνει τον αριθμό των McNuggets σε κάθε είδος πακέτου.

OUTPUT FORMAT

Το αρχείο output πρέπει να έχει μία μόνο γραμμή που θα περιέχει έναν μοναδικό ακέραιο αριθμό, ο οποίος παριστάνει τον μεγαλύτερο αριθμό από nuggets που δεν μπορούν να αγοραστούν ή 0 αν όλες οι δυνατές αγορές μπορούν να γίνουν ή δεν υπάρχει όριο για τον μεγαλύτερο αριθμό.

SAMPLE INPUT (file INPUT.TXT):

```
3
3
6
10
```

SAMPLE OUTPUT (file OUTPUT.TXT):

```
17
```

Πρόβλημα 4: Ο φράχτης [Brian Dean]

Οι φράχτες που περιφράσουν τον βοσκότοπο του μπάρμπα Στάθη έχουν ξεφύγει από κάθε έλεγχο. Είναι φτιαγμένοι από ευθύγραμμα κομμάτια τα οποία ενώνονται μεταξύ τους μόνο στα άκρα τους. Κάποιες φορές υπάρχουν περισσότεροι από δύο φράχτες που ενώνονται στο ίδιο σημείο. Το αποτέλεσμα είναι ένα δίκτυο από φράχτες που περικλείουν τα χωράφια. Ο μπάρμπα Στάθης θέλει να αρχίσει να φέρνει "βόλτα" τα πράγματα. Για την ακρίβεια, θέλει να ξέρει ποιο από τα χωράφια έχει τη μικρότερη περίμετρο.

Ο μπάρμπα Στάθης έχει αριθμήσει τα κομμάτια του φράχτη από 1 μέχρι N (N = ο συνολικός αριθμός από κομμάτια). Ξέρει επίσης τα παρακάτω για κάθε κομμάτι:

το μήκος του κάθε κομματιού

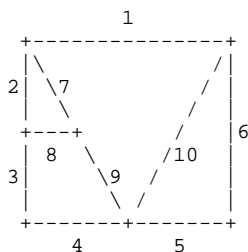
τα κομμάτια με τα οποία ενώνεται στο ένα άκρο του

τα κομμάτια με τα οποία ενώνεται στο άλλο άκρο του

Οποσδήποτε, κανένα κομμάτι δεν ενώνεται με τον εαυτό του.

Σας δίνεται μία λίστα από κομμάτια του φράχτη, τα οποία δείχνουν ένα σύνολο από χωράφια. Γράψτε ένα πρόγραμμα που να υπολογίζει τη μικρότερη περίμετρο χωραφιού. Σαν παράδειγμα, σκεφτείτε την παρακάτω κατασκευή, με κομμάτια από 1 έως 10:

Για παράδειγμα, τα χωράφια μπορούν να είναι όπως αυτά (οι αριθμοί είναι τα ID των κομματιών):



Το χωράφι με τη μικρότερη περίμετρο είναι αυτό που περικλείεται από τα κομμάτια 2, 7 και 8.

INPUT FORMAT

Η πρώτη γραμμή του αρχείου input περιέχει έναν μόνο ακέραιο, N ($1 \leq N \leq 100$).

Το υπόλοιπο αρχείο input περιέχει N ομάδες από εγγραφές, κάθε μία των τριών γραμμών.

Η πρώτη γραμμή από κάθε εγγραφή περιέχει τον αριθμό του κομματιού ($1 \leq s \leq N$), το μήκος του ($1 \leq \text{μήκος} \leq 255$), τον αριθμό των κομματιών της επόμενης γραμμής ($N1$), και τον αριθμό των κομματιών της μεθεπόμενης γραμμής ($N2$).

Η δεύτερη γραμμή της κάθε εγγραφής περιέχει $N1$ ($1 \leq N1 \leq 8$) ακεραίους, κάθε ένας δηλώνει τον αριθμό του κομματιού που ενώνεται με το παρόν στο ένα άκρο. Η τρίτη γραμμή περιέχει $N2$ ($1 \leq N2 \leq 8$) ακεραίους, που είναι τα κομμάτια που ενώνονται με το παρόν στο άλλο άκρο.

OUTPUT FORMAT

Το αρχείο output θα περιέχει μία μόνο γραμμή με έναν μόνο ακέραιο αριθμό, ο οποίος δείχνει την μικρότερη περίμετρο χωραφιού.

SAMPLE INPUT (file INPUT.TXT):

```

10
1 16 2 2
2 7
10 6
2 3 2 2
1 7
8 3
3 3 2 1
8 2
4
4 8 1 3
3
9 10 5
5 8 3 1
9 10 4
6
6 6 1 2
5
1 10
7 5 2 2
1 2
8 9
8 4 2 2
2 3
7 9
9 5 2 3
7 8
4 5 10
10 10 2 3
1 6
4 9 5

```

SAMPLE OUTPUT (file OUTPUT.TXT):

```

12

```

USACO 98 Spring Championship

Problem 1: DIFFERENT DICE [Hal Burch]

In a different galaxy far far away there are different kinds of dice, often with unusual integer numbers of sides. These dice can have anywhere from 2 to 16 (integer) sides. Each side can have from 1 to 32 (integer) spots. Sets of dice contain anywhere from 1 through 32 dice.

These dice roll just like Earth-based dice, of course: the outcome of rolling is considered to be the sum of the spots showing on each die.

Anthropologists digging through old ruins in the galaxy far far away have found sets of dice that have various different numbers of spots. Here's one set of two two-sided dice they found:

* {2,3} and {3,4}

and here's another:

* {1,2} and {4,5}

It's easy to see that the first set of dice yields sums of 5, 6, and 7. The second set also yields 5, 6, and 7. Furthermore, the probability of rolling a five is 1/4, rolling a six is 1/2, and rolling a seven is 1/4 for both sets of dice.

Given two sets of dice, your program must decide if they yield the same sets of sums. Additionally, it must also decide if the two sets yield those sums with the same probabilities.

INPUT FORMAT:

The first line of the input file contains two space-separated integers D1 and S1. D1 is the number of dice in the first set, and S1 is the number of sides on each die.

The subsequent D1 lines each contain S1 values that specify the spot values for a die. The values might or might not all be different.

The next line of the input file contains two space-separated integers D2 and S2. D2 is the number of dice in the second set, and S2 is the number of sides on each die.

The subsequent D2 lines each contain S2 values that specify the spot values for a die. The values might or might not all be different.

OUTPUT FORMAT:

The first line of the output file contains a 'Y' or 'N'. 'Y' indicates that both sets of dice yield the same set of possible sums. 'N' indicates otherwise.

The second line of the output file contains a 'Y' or 'N'. 'Y' indicates that both sets of dice yield the same set of possible sums with the same set of probabilities of achieving those sums. 'N' indicates otherwise.

The third line of output contains two, space-separated integers. The first integer indicates the number of possible sums the first set of dice could yield. The second integer indicates the number of different sums the second set of dice could yield.

SAMPLE INPUT (file INPUT.TXT):

```
2 6
1 2 3 4 5 6
1 2 4 3 4 1
3 4
1 4 3 2
5 3 2 1
1 1 1 1
```

SAMPLE OUTPUT (file OUTPUT.TXT):

```
N
N
9 8
```

Problem 2: FEED RATIOS [Similar to ACM Finals, 1998, forwarded by Dan Adkins]

Farmer John feeds his cows only the finest mixture of cow food, which has three components: Barley, Oats, and Wheat. While he knows the precise mixture of these easily mixable grains, he can not buy that mixture! He buys three other mixtures of the three grains and then combines them to form the perfect mixture.

Given a set of integer ratios barley:oats:wheat, find a way to combine them IN INTEGER MULTIPLES to form a mix with some goal ratio x:y:z.

For example, given the goal:

3:4:5

and the ratios of three mixtures:

1:2:3

3:7:1

2:1:2

Your program should find some minimum number of integer units (the 'mixture') of the first, second, and third mixture that should be mixed together to achieve the goal ratio or print 'NONE'. 'Minimum number' means the sum of the three non-negative mixture integers is minimized.

For this example, you can combine eight units of mixture 1, two units of mixture 2, and five units of mixture 3 to get seven units of the goal ratio:

$$8*(1:2:3) + 1*(3:7:1) + 5*(2:1:2) = (21:28:35) = 7*(3:4:5)$$

Integers in the goal ratio and mixture ratios are all non-negative and smaller than 100 in magnitude.

INPUT FORMAT:

The first line of the input file contains three space separated integers that represent the value of the goal ratios.

The next three lines of the input file each contain three space separated integers that represent the ratios of the three mixtures purchased.

OUTPUT FORMAT:

The output file should contain one line containing four integers or the word 'NONE'. The first three integers should represent the number of units of each mixture to use to obtain the goal ratio. The fourth number should be the multiple of the goal ratio obtained by mixing the initial feed using the first three integers as mixing ratios.

SAMPLE INPUT (file INPUT.TXT):

```
3 4 5
1 2 3
3 7 1
2 1 2
```

SAMPLE OUTPUT (file OUTPUT.TXT):

```
8 1 5 7
```


Problem 3: THE TAMWORTH TWO [Richard Forster, BOI]

A pair of cows is loose somewhere in the forest. Farmer John is lending his expertise to their capture. Your task is to model their behavior.

The chase takes place on a 10 by 10 planar grid. Squares can be empty or they can contain: an obstacle, the cows (who always travel together), or Farmer John. The cows and Farmer John can occupy the same square (when they `meet') but neither the cows nor Farmer John can share a square with an obstacle.

Each square is represented as follows:

```
. Empty square
* Obstacle
C Cows
F Farmer
```

Here is a sample grid:

```
*...*.....
.....*....
...*...*...
.....
...*.F.....
*.....*....
...*.....
..C.....*
...*...*...
.*...*.....
```

The cows wander around the grid in a fixed way. Each minute, they either move forward or rotate. Normally, they move one square in the direction they are facing. If there is an obstacle in the way or they would leave the board by walking `forward', then they spend the entire minute rotating 90 degrees clockwise.

Farmer John, wise in the ways of cows, moves in exactly the same way.

The farmer and the cows can be considered to move simultaneously during each minute. If the farmer and the cows pass each other while moving, they are not considered to have met. The chase ends when Farmer John and the cows occupy the same square at the end of a minute.

Read a ten-line grid from INPUT.TXT that represents the initial state of the cows, Farmer John, and obstacles. Each of the ten lines contains exactly ten characters using the coding above. There is guaranteed to be only one farmer and one pair of cows. The cows and Farmer John will not initially be on the same square.

Calculate the number of minutes until the cows and Farmer John meet. Assume both the cows and farmer begin the simulation facing in the `up' direction. Print 0 if they will never meet.

INPUT FORMAT:

The input file contains ten lines of ten characters (no spaces).

OUTPUT FORMAT:

The output file should contain one line containing a single integer that represents the numbers of minutes until Farmer John and the Cows meet. Print 0 if they never meet.

SAMPLE INPUT (file INPUT.TXT):

```
*...*.....
.....*....
...*...*...
.....
...*.F.....
*.....*....
```

```

...*.....
..C.....*
...*.....
.*.....

```

SAMPLE OUTPUT (file OUTPUT.TXT):

49

Problem 4: SUBSET SUMS [Recreational Mathematics Newsletter, Kolstad]

For many sets of consecutive integers from 1 through N ($1 \leq N \leq 39$), one can partition the set into two sets whose sums are identical.

For example, if $N=3$, one can partition the set $\{1, 2, 3\}$ in one way so that the sums of both subsets are identical:

$\{3\}$ and $\{1,2\}$

This counts as a single partitioning (i.e., reversing the order counts as the same partitioning and thus does not increase the count of partitions).

If $N=7$, there are 4 ways to partition the set $\{1, 2, 3, \dots, 7\}$ so that each partition has the same sum:

$\{1,6,7\}$ and $\{2,3,4,5\}$
 $\{2,5,7\}$ and $\{1,3,4,6\}$
 $\{3,4,7\}$ and $\{1,2,5,6\}$
 $\{1,2,4,7\}$ and $\{3,5,6\}$

Given N, your program should print the number of ways a set containing the integers from 1 through N can be partitioned into two sets whose sums are identical. Print 0 if there are no such ways.

Your program must calculate the answer, not look it up from a table.

INPUT FORMAT:

The input file contains a single line with a single integer representing N, as above.

OUTPUT FORMAT:

The output file contains a single line with a single integer that tells how many same-sum partitions can be made from the set $\{1, 2, \dots, N\}$. The output file should contain 0 if there are no ways to make a same-sum partition.

SAMPLE INPUT (file INPUT.TXT):

7

SAMPLE OUTPUT (file OUTPUT.TXT):

4

Problem 5: STRINGSOBITS [Kim Schrijvers]

Consider an ordered set S of strings of N ($1 \leq N \leq 32$) bits. Bits, of course, are either 0 or 1.

This set of strings is interesting because it is ordered and contains all possible strings of length N that have L ($1 \leq L \leq N$) or fewer bits that are '1'.

Your task is to read a number I ($1 \leq I \leq \text{sizeof}(S)$) from the input and print the Ith element of the ordered set.

INPUT FORMAT:

The input file contains a single line with three space-separated integers, N, L, and I.

OUTPUT FORMAT:

The output file should contain a single line with a single integer that represents the Ith element of the set of integers of length N bits with no more than L bits that are `1'.

SAMPLE INPUT (file INPUT.TXT):

5 3 19

SAMPLE OUTPUT (file OUTPUT.TXT):

10100