ECOO 2000
Programming Contest

Boardwide Contest

Problem 1 – Time

At any given time, the hour hand and the minute hand of a clock will form an angle. Write a program that calculates and prints the angle in degrees, minutes and seconds (rounded), given the time in hours, minutes and seconds. Note that the angle is always 180^o or less.

Note that a degree measure is divided into 60 minutes or 3600 seconds.

Your program will read from an input file (**DATA11.txt** for the first try and **DATA12.txt** for the second) five lines. Each line contains 8 characters representing the time in the form hh:mm:ss

Print out the data on a cleared screen (window) Each output must be separated from the others by a blank line and be of the form: ddd:mm:ss.

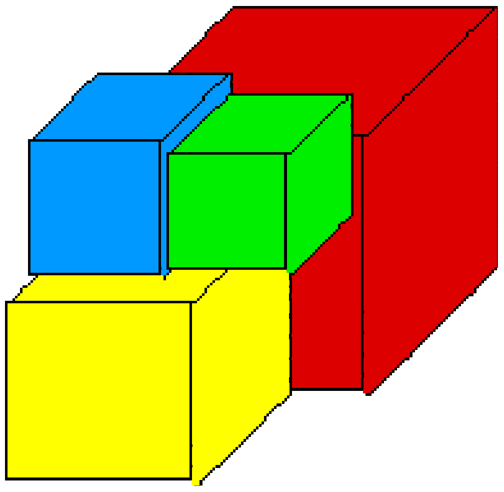
Sample input

```
   12:00:00
   06:00:00
   03:30:00
   12:25:05
   06:00:10
```

Sample output

```
  000:00:00
  180:00:00
  075:00:00
  137:57:30
  179:05:00
```

Problem 2 – Four Cubes



A packing company needs an economical way in which to package sets of four cubes. These cubes come in different sizes, and the company has a packaging machine that will create appropriate sized boxes, given the three dimensions.

Your task is to write a program that will find the smallest volume of box in which the four cubes can fit. (The empty spaces around the cubes will be filled with styrofoam).

Your program will read from an input file (DATA21.txt for the first try and DATA22.txt for the second), 5 sets of 4 integers. The 4 numbers represent the sides of the 4 cubes, its values between 1 and 1024

Print the three dimensions of the packing box in decreasing order (the largest side first, the smallest side last) all on one line as is shown in the example below.

If more than one arrangement will yield the same minimum volume, than any one solution is valid: (if all 4 cubes are the same size, with say 10 cm to the side, then the minimum value of 4000 cm3 can be obtained by a box of dimensions: 40x10x10 and a box of dimensions: 20x20x10)

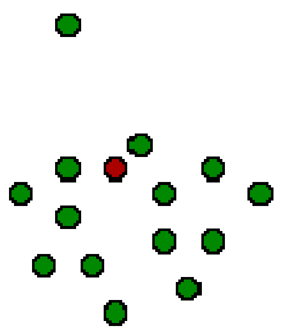
Sample Input

```
10 30 50 5
10 10 20 20
6 7 9 8
1 2 3 4
67 88 3 90
```

Sample Output

```
80x50x50 with a volume of: 200000
50x20x20 with a volume of: 20000
17x15x9 with a volume of: 2295
7x5x4 with a volume of: 140
245x90x90 with a volume of: 1984500
```

Problem 3 – Radio Station



A radio station wishes to find the one place in town that is central to the population of a given town. It means that it wants to minimize the distance to each and every resident.

Its strategy is to find the two houses that are furthest apart from each other, and position itself halfway between them. It is not the perfect solution, but the station owner finds that it is good enough. Based on all the data given, the station will create a map, where each residence is represented by a green circle, and the radio station by a red one.

First, based on some arbitrary origin, it assigns an ordered pair of integers (x,y) to the position of each residence . It then proceeds to find its own future location.

The input file **DATA31.txt** (**DATA32.txt** for the second attempt) contains 5 sets of data. The first line contains the number of houses (n), and the next n lines contain the x and y values (integers) of the positions of n residences, with one space in between.

Your output will be five "maps" of residences (represented by green circles) and one radio station (represented by a red circle).

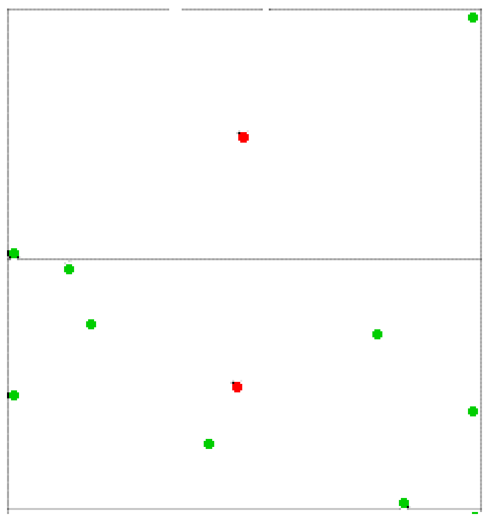
Each of the 5 maps must remain on the screen until any key is pressed.

Each map must be scaled along x and y in such a way, that it occupies all available space of the output screen. This means that the circle representing the residence with the greatest x-value must touch or nearly touch the right hand edge of the screen; the one with the largest y-value the top of the screen; the one with the smallest x-value the left hand edge of the screen; the one with the smallest y-value the lowest edge of the screen. The other circles will be placed on the screen in proportion.

Input (two of five sets)

```
3
200 200
100 100
300 300
7
34 59
90 6
100 51
42 120
62 35
45 93
86 88
```

Output



Problem 4 – Fractions

Imagine that your little brother has to do a series of multiplication and division problems for homework. His teacher created a program that will generate a series of random fractions, and is now emailing innocent little eleven-year-olds, like your brother, with huge amounts of homework.

The problems consist of a set of up to 5 fractions each that have to be multiplied or divided.
He is turning to you, with the hope that you can quickly create a program that will generate a solution to all of them.

In order to test you program, **DATA41.txt** (**DATA42.txt** for the second try) consists of 5 lines of data containing up to 5 fractions each. They are separated by either a * (for multiplication) or a / (for division)

Fractions will always be between round brackets such as (-5/8) and (4/1) or (4) Note from the example that there may be extra spaces in the line. Note also, that individual integers, including those in the answers, are not larger than 2^{31}.

Output should be on a cleared screen (or window). The five lines should be separated by a blank line as shown in the example below. Fractions in the answer must be simplified, and a denominator of 1 avoided. Only the numerator may have a negative sign. Each operator must be surrounded by exactly one space, and no other spacing is allowed.

Sample input

```
(15) * (15/12)/(4/5)
(4/16)   /(12/5)*(7/2)/(-49/8)/(3)
(5/1)
(256/7)/(8/49)/(7/8)/(6)
(1)*(1)/(1/6)
```

Sample output

```
(15) * (15/12) / (4/5) = (375/16)
(4/16) / (12/5) * (7/2) / (-49/8) / (3) = (-5/252)
(5) = (5)
(256/7) / (8/49) / (7/8) / (6) = (128/3)
(1) * (1) / (1/6) = (6)
```

ECOO 2000
Programming Contest

Regional Contest

Problem 1 – The Game of Q

A certain crossword game consist of a 3x3 square, and two players. Each player alternates placing letters in the square, on the roll of a die. For the purposes of this problem, you do not need to know that on an even throw, a person may only place a vowel and on an odd throw a consonant. Only by rolling a 1 can one place the letter "Q", and each letter can be used only once.

One player tries to create vertical 3-letter words and the other player tries to create horizontal 3-letter words.
Each consonant in a valid word is 1 point, a vowel 2, and the letter Q=3
However, a 3-letter word is only valid, if it obeys the following 7 rules:

1. The word must contain at least one vowel (A, E, I, O, U)
   (the letter Y is considered a consonant)
2. The word must contain at least one consonant
3. If a word starts with a vowel and ends in a consonant, the middle letter must be a vowel or one of: L , R
4. If a word ends with a vowel and starts with a consonant, the middle letter must be a vowel or one of: L , S
5. The word may not end with a V,W or Y
6. The word may not start with an X
7. The Q must be followed by a U (middle letter) and one other vowel

Write a program that will read from a text file (DATA11.txt for the first try and DATA12.txt for the second) 5 sets of data. Each set of data consists of 3 lines containing a 3-letter word in capital letters. This data represents the completed 3x3 square, and it is your task to find the correct score.

Output 10 lines on a cleared screen. Each set of two lines must be separated by a blank line and indicate the score for "Horizontal" and "Vertical", as in the example below. The individual marks are for words from left to right or from top to bottom.

Sample Input

```
GQE OUT PAI
PLO SRP AEI
KTL QUR MEW
VIE LUR OMA
TIF ESV PLO
```

Sample Output

```
Horizontal score: 0 + 5 + 5 = 10
Vertical score:   4 + 7 + 5 = 16

Horizontal score: 4 + 0 + 0 =  4
Vertical score:   4 + 0 + 5 =  9

Horizontal score: 0 + 0 + 0 =  0
Vertical score:   0 + 5 + 0 =  5

Horizontal score: 5 + 4 + 5 = 14
Vertical score:   4 + 5 + 5 = 14

Horizontal score: 4 + 0 + 4 =  8
Vertical score:   4 + 0 + 0 =  4
```

Problem 2 – Matrix Multiplication

Matrices are formed of numbers arranged in rectangles.

Two matrices can be multiplied by taking rows of the left matrix and columns of the right matrix, multiplying the first element of a row with the first element of a column, the second element of a row with the second element of the column, and so on. The products are then added. The new matrix is formed of all the possible combinations of results. the x-th row of the left matrix multiplied by the y-th column of the right matrix becomes the new element in the x-th row and y-th column.

Clearly, the number of columns of the left matrix must match the number of rows of the right matrix. And a 5x3 matrix multiplied by a 3x4 matrix will result in a 5x4 matrix:

$$\begin{bmatrix} 5 & 2 & 1 \\ \textcircled{8} & \textcircled{4} & \textcircled{12} \\ -5 & 6 & 3 \\ 3 & 2 & -12 \\ 2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} -9 & 8 & \textcircled{13} & -6 \\ 5 & -1 & \textcircled{2} & 3 \\ -2 & 3 & \textcircled{8} & 13 \end{bmatrix} - \begin{bmatrix} -37 & 35 & 77 & -11 \\ -76 & 84 & \textcircled{208} & 120 \\ 60 & -55 & -20 & 87 \\ 7 & -20 & -53 & -168 \\ 5 & -8 & 0 & -55 \end{bmatrix}$$

And consider for example the 2nd row X 3rd column, resulting in the element at row2, column 3, that is: (8 X 13) + (4 X 2) + (12 X 8) = 208

Write a program that will read from a text file (DATA21.txt for the first try and DATA22.txt for the second) 5 sets of data. The first line of each set contains the number of rows of the left matrix (x), the number of columns of the first matrix (y), which also denote the number of rows of the right matrix and the number of columns of the right matrix (z). The next x lines contain the rows of data of the left matrix, followed by y lines containing the rows of data of the right matrix. Each set of row entries is bracketed by angle brackets as in the sample on the turn page.

Note that x,y,z are numbers between 1 and 6 inclusive, and that the total number of columns of the 3 matrices will never exceed 16.

Print the multiplication problem for each of the 5 sets separately on a cleared screen (or window). Create a pause between the output of each set, so that you can proceed only after pressing any key.

Notice also, that the centres of matrices must be aligned with the **X** sign and the = sign

For the brackets, you might use the standard IBM font set and chr(218), chr(179), chr(192) for the opening bracket chr(191), chr(179), chr(217) for the closing bracket (if that is not available to you, use the characters "/", "|" and "\" instead)

Sample input

```
6 5 2
<5 2 1 0 -3>
<8 4 12 -6 2>
<-5 6 3 1 -4>
<3 2 -12 7 5>
<0 2 3 -4 -23>
<-9 8 13 -6 6>
<5 -4>
<-2 3>
<0 9>
<0 -13>
<-8 1>
5 3 4
<5 2 1>
<8 4 12>
<-5 6 3>
<3 2 -12>
<2 3 -4>
<-9 8 13 -6>
<5 -4 2 3>
<-2 3 8 13>
6 5 5
<5 2 1 0 -3>
<8 4 12 -6 2>
<-5 6 3 1 -4>
<3 2 -12 7 5>
<0 2 3 -4 -23>
<-9 8 13 -6 6>
<5 -4 6 3 1 -4>
<-2 3 3 2 -12 9>
<0 9 -2 3 8 13>
<0 -13 4 12 -6 2>
<2 -12 7 5 2 -12>
1 6 1 <5 2 1 0 -3 2>
<8>
<4>
<12>
<-6>
<-5>
<6>
6 2 4
<5 -5>
<2 3>
<1 2>
<0 13>
<-3 -9>
<8 -4>
<4 12 -6 2>
<-4 6 1 3>
```

$$\begin{bmatrix} 5 & 2 & 1 & 0 & -3 \\ 8 & 4 & 12 & -6 & 2 \\ -5 & 6 & 3 & 1 & -4 \\ 3 & 2 & -12 & 7 & 5 \\ 0 & 2 & 3 & -4 & -23 \\ -9 & 8 & 13 & -6 & 6 \end{bmatrix} \text{X} \begin{bmatrix} 5 & -4 \\ -2 & 3 \\ 0 & 9 \\ 0 & -13 \\ -8 & 1 \end{bmatrix} = \begin{bmatrix} 45 & -8 \\ 16 & 168 \\ -5 & 48 \\ -29 & -200 \\ 180 & 62 \\ -109 & 261 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 2 & 1 \\ 8 & 4 & 12 \\ -5 & 6 & 3 \\ 3 & 2 & -12 \\ 2 & 3 & -4 \end{bmatrix} \text{X} \begin{bmatrix} -9 & 8 & 13 & -6 \\ 5 & -4 & 2 & 3 \\ -2 & 3 & 8 & 13 \end{bmatrix} = \begin{bmatrix} -37 & 35 & 77 & -11 \\ -76 & 84 & 208 & 120 \\ 69 & -55 & -29 & 87 \\ 7 & -20 & -53 & -168 \\ 5 & -8 & 0 & -55 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 2 & 1 & 0 & -3 \\ 8 & 4 & 12 & -6 & 2 \\ -5 & 6 & 3 & 1 & -4 \\ 3 & 2 & -12 & 7 & 5 \\ 0 & 2 & 3 & -4 & -23 \\ -9 & 8 & 13 & -6 & 6 \end{bmatrix} \text{X} \begin{bmatrix} 5 & -4 & 6 & 3 & 1 \\ -2 & 3 & 3 & 2 & -12 \\ 0 & 9 & -2 & 3 & 8 \\ 0 & -13 & 4 & 12 & -6 \\ 2 & -12 & 7 & 5 & 2 \end{bmatrix} = \begin{bmatrix} 15 & 31 & 13 & 7 & -17 \\ 36 & 142 & 26 & 6 & 96 \\ -45 & 100 & -42 & -2 & -67 \\ 21 & -205 & 111 & 86 & -149 \\ -50 & 361 & -177 & -150 & -22 \\ -49 & 183 & -38 & -14 & 47 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 2 & 1 & 0 & -3 & 2 \end{bmatrix} \text{X} \begin{bmatrix} 8 \\ 4 \\ 12 \\ -6 \\ -5 \\ 6 \end{bmatrix} = \begin{bmatrix} 87 \end{bmatrix}$$

$$\begin{bmatrix} 5 & -5 \\ 2 & 3 \\ 1 & 2 \\ 0 & 13 \\ -3 & -9 \\ 8 & -4 \end{bmatrix} \text{X} \begin{bmatrix} 4 & 12 & -6 & 2 \\ -4 & 6 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 40 & 30 & -35 & -5 \\ -4 & 42 & -9 & 13 \\ -4 & 24 & -4 & 8 \\ -52 & 78 & 13 & 39 \\ 24 & -90 & 9 & -33 \\ 48 & 72 & -52 & 4 \end{bmatrix}$$

Problem 3 – Matrix Multiplication

Picture a text screen containing outlines of shapes. Since this is a text screen, the outlines are formed out of characters. It is your task, as the programmer, to fill these shapes with the same character as the outline (see the example below).

There will be five shapes: A shape enclosed by A's, a shape enclosed by "B"'s, and so on. The last shape is enclosed by the letter "E". In some cases a shape ends at the edge of the text screen. Note from the example below, that a **diagonal** line may separate the **inside** from the **outside** of a shape.

The small letter "x" in the input file represents empty space. On the screen you must replace those letters with blanks.

For your convenience, exactly one letter **P** is placed inside each shape.

The first line of the input file (DATA31.txt for the first try and DATA32.txt for the second) contains the width of the text rectangle and the second line contains its length (i.e. the number of lines of text that follow) In all cases, the text rectangle will fit a 79x25 text screen

Print the text rectangle on a cleared screen, showing only the filled shapes:

Sample Input

```
34 25
xxxxAAAxxxPxAxxxxxxxxxxxxExxxxxxxx
xxxAxxxxxxxxAxxxxxxxxxxxxExxxxxxxx
xxxAxxAxxxAAxxxxxxxxxxxxxExxxPxxxx
xxAxxAxAAAxxxxxxxxxxxxxxxExxxxxxxx
xAxxxAxxxxxxxxxxxxxxxxxxxEEEEEEEEE
xxAxxAxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxAAAxxxxxxxxxxxxxxxxxxxCCCxxxxxxx
xxxxxxxxxxxxxxxxxxCCCCCCxxxCCxxxxx
xxxxxxxxxxxxBBBBBCxxxxxPxxxxCxxxxx
xxxxxxxxxxBBxxxxxBCCCCCCCCCCCxxxxx
xxxxxxxxxxxBBxxxBxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxBBBxxBxxxxxxxxxxxxxxxx
xxxxxxxxxxxBxxBxxxBxxxxxxxxxxxxxxx
xxxxxxxxxxBxxxBxxxxBxxxxxxxxxxxxxx
BBBBBBBxxBxxxxPxxxxxBxxxxxxxxxxxxx
xxxxxxxBBxxxxxxxxBBBxxxxxxxxxxxxxx
xxxxxxxxxxBBBBBBBxxxxxxxxxxxxxxxxx
BBBBBBBBBBxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxDDDDDDDxxxxxxxxxxxx
xxxxxxxxxxxxxxxDxxxxxxDDxxxxxxxxxx
xxxxxxxxxxxxxxDxxxxxxxxxDDDDDDDDDD
xxxxxxxxxxxxxDxxxxxPxxxxxxxxxxxxxx
xxxxxxxxxxxxDxxxxxxxxxxxxxxxxxxxxx
```

Sample Output

```

    AAAAAAAAA            EEEEEEEEE
   AAAAAAAAAA            EEEEEEEEE
   AAAAAAAAA             EEEEEEEEE
  AAAA AAA               EEEEEEEEE
 AAAAA                   EEEEEEEEE
  AAAA
  AAA                   CCC
                  CCCCCCCCCCC
            BBBBBCCCCCCCCCCCC
          BBBBBBBBCCCCCCCCCCC
           BBBBBB
            BBBBBB
           BBBBBBBB
          BBBBBBBBBB
BBBBBBB  BBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBB
BBBBBBBBBB


                           DDDDDDD
                         DDDDDDDDD
              DDDDDDDDDDDDDDDDDDDD
             DDDDDDDDDDDDDDDDDDDDD
            DDDDDDDDDDDDDDDDDDDDDD
```

Problem 4 – Drop Box

At a certain college, students drop their homework off in the drop box of their Computer Science professor. His two graduate students will mark these tests as they come in. If, however, both students are still busy with marking previously handed in work, the homework stays in the box until they can get to it. It may happen that several students drop off their work before they can be marked. It is your task to write a program that will find out who will mark whose test, and in what order. Each marker takes his or her own time marking each test. The marker who marks fastest, is also fastest picking up a paper. So, if both are waiting for a paper to arrive, the fastest marker gets the paper.

For example, 5 students hand in their marks as follows:
Markers Mohammed and Fatima will be marking. Mohammed takes 31 minutes for each paper and Fatima 30 minutes.
Papers will be dropped in as follows: Mary at 11:41, followed by: John at 11:43. Nora at 11:45, Tony at 11:59 and Olivia at 12:08.
Fatima is the fastest, and so:
Fatima marks Mary's paper from 11:41 to 12:11.
Mohammed marks John's paper from 11:43 to 12:14
meanwhile Nora, then Tony and finally Olivia drop off their papers.
At 12:11 Fatima picks up the top paper (Olivia's) and marks it until 12:41
Mohammed picks up the next paper (Tony's) at 12:14 and marks it until 12:45
Finally Fatima picks up the last paper in the drop box, Nora's.

The input file (DATA41.txt for the first try and DATA42.txt for the second) consists of 5 sets of data. The first line contains the total number of papers to be marked. The second line contains the name of one marker, a space, and the number of minutes that person takes for each paper. The third line contains the two pieces of data (name and number of minutes) for marker number 2. The remaining lines contain two data each separated by one space: a name and a time.

All names consist of a set of letters, no more than 15 characters long, and no embedded spaces. There are between 3 to 60 papers to be marked in each set. The time will be between 9:00 and 17:00 inclusive (h:mm or hh:mm) and no two students will deposit their papers at the same time.

Output on a cleared screen the names of the markers followed by one or more columns of names, that are to be marked by these graduate students. (Each column may contain no more than 20 names)

Note that the names of students with their times are not necessarily in the right order.

Sample Input
(3 sets only)

```
20
Sally 16
Mary 18
Moe 11:23
Darryl 9:47
Gerry 10:32
Randy 12:10
Val 12:40
Harry 10:33
Evelyn 10:10
Heather 10:43
Tony 12:38
Boris 9:32
Perry 11:50
Claude 9:45
Garry 10:31
Ian 10:49
Olivia 11:44
Drew 9:57
Kelly 11:02
Andy 9:13
Sally 12:20
Fred 10:23
17
Sam 11
Tim 8
Randy 12:36
Amy  9:08
Bert 9:23
Garry 10:34
Pete 12:28
Fred 10:24
Gerry 10:43
Claude 9:54
Orville 12:10
Quinton 12:33
Evelyn 10:09
Chris 9:45
Perry 12:19
Jason 11:10
Veronica 13:20
Andy 9:18
Moe  11:49
5
Mohammed 31
Fatima 30
John 11:43
Nora 11:45
Tony 11:59
Mary 11:41
Olivia 12:08
```

Sample Ouput

```
Sally                              Mary

Andy                               Claude
Boris                              Drew
Darryl                             Fred
Evelyn                             Harry
Garry                              Ian
Heather                            Gerry
Kelly                              Perry
Moe                                Sally
Olivia                             Val
Randy
Tony
```

```
 Tim                               Sam

 Amy                               Bert
 Andy                              Quinton
 Chris
 Claude
 Evelyn
 Fred
 Garry
 Gerry
 Jason
 Moe
 Orville
 Perry
 Pete
 Randy
 Veronica
```

```
Fatima                             Mohammed

Mary                               John
Olivia                             Tony
Nora
```

ECOO 2000

Programming Contest

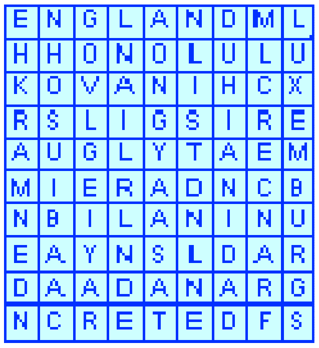
Final Contest

Problem 1 – Word Puzzle

ENGLAND
HOLLAND
BELGIUM
ITALY
CANADA
GRANADA
LUXEMBURG
HONOLULU
CRETE
DENMARK
CHINA
FRANCE
SYRIA
US

Imagine a rectangular array letters in which are hidden a list of words. In many of these word games you must eliminate all the words in order to discover the leftover letters, which in turn form the hidden message.

Figure 1, for example, is a 9 by 10 array of letters in which are hidden the names of 14 countries (see left)



E	N	G	L	A	N	D	M	L
H	H	O	N	O	L	U	L	U
K	O	V	A	N	I	H	C	X
R	S	L	I	G	S	I	R	E
A	U	G	L	Y	T	A	E	M
M	I	E	R	A	D	N	C	B
N	B	I	L	A	N	I	N	U
E	A	Y	N	S	L	D	A	R
D	A	A	D	A	N	A	R	G
N	C	R	E	T	E	D	F	S

fig. 1

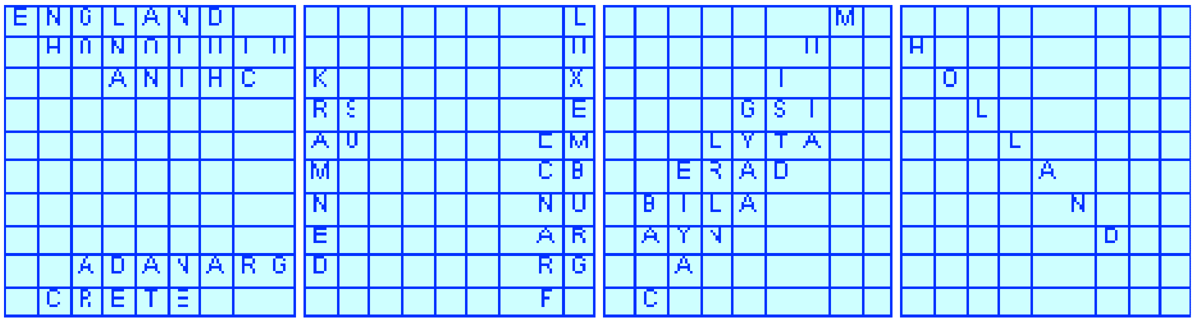


fig. 2

These letters are arranged horizontally, vertically and diagonally, as shown in fig. 2. When these names are eliminated, the remaining letters form the name "VIRGINISLANDS" (see fig. 3). The text file data11.txt (data12.txt for the second try) will contain five sets of data. Each set of data consist of information on the rectangle and a list of words.

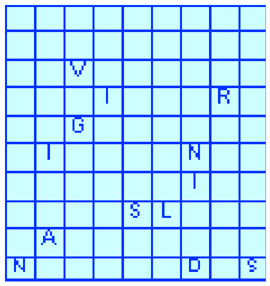


fig. 3

The first two lines of the set of data contain the number of columns (c) and the number of rows (r), followed by r lines containing the rectangle of letters. This is followed by a line containing the number of words in the list (n), and n lines containing the words of the list.

It is your task to find the hidden message in the remaining letters, when all the words have been eliminated.

Print out each hidden message on a cleared screen, separated by blank lines. See for example the sample input and output data on the next page.

Sample Input (3 sets only)

```
8 7
THMAVISE
STANLEYF
TIRROWAN
ELYMOERP
VSATPLEA
EUNZZTLT
NONAEERS
12
HAROLD
STANLEY
MOE
STEVEN
MARYANN
MAVIS
NONA
PETE
LEA
PAT
ROWAN
ELY
4 3
FTGI
BOAR
DSHC
3
BOAR
DOG
CAT
9 10
ENGLANDML
HHONOLULU
KOVANIHCX
RSLIGSIRE
AUGLYTAEM
MIERADNCB
NBILANINU
EAYNSLDAR
DAADANARG
NCRETEDFS
14
ENGLAND
HOLLAND
BELGIUM
ITALY
CANADA
GRANADA
LUXEMBURG
HONOLULU
CRETE
DENMARK
CHINA
FRANCE
SYRIA
US
```

Sample Ouput

```
THEFIRSTPUZZLERS

FISH

VIRGINISLANDS
```

Problem 2 – Projects

Complex projects must be organized in smaller tasks that can then be placed in chronological order. Morely needs help organizing his project, and is turning to you for help.

You are asked to write a program that can sort up to 10 tasks (A,B, ... , J) based on up to 10 clues. If there are more than one solution to the problem, then any one will do.

Clues are of the form illustrated below:
B:FGHC means that task B depends on tasks F, G, H and C and must therefore be performed after all these other tasks have been done.

Information will come from DATA21.txt (DATA22.txt for the second try). The text file will contain 5 sets of data: The first line of each set contains the number of tasks. (tasks are named by the appropriate letter of the alphabet: If there are 5 tasks to be performed, they would be named A. B, C, D and E) The second line will contain the number of clues that follow.
A clue consists of one line composed of one letter, a colon, and up to 9 other letters, as shown in the example below. There are no embedded blanks.

All task relationships are consistent, and resolvable. Print the tasks in order in which they should be performed, as shown in the example below:

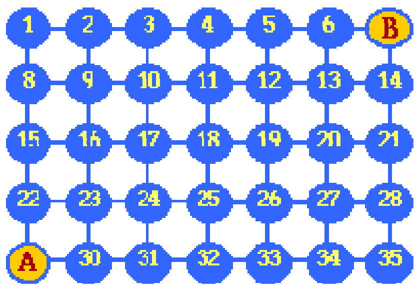
Sample Input (3 sets only)

```
4
3
D:C
C:A
B:DC
10
10
A:BDI
F:IAEG
J:HIDB
E:GBD
C:F
B:GI
D:B
J:CI
H:C
G:I
2
1
A:B
```

Sample Output

```
ACDB
IGBDEAFCHJ
BA
```

Problem 3 – Truck Driver

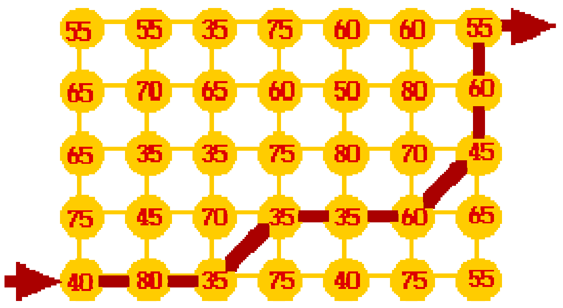


A truck driver wishes to go from city A to city B, along the grid as shown. She can only move east (right) and north (up).
The numbers represent cities, and moving from one city to the next represents a one day trip.
It would therefore take a ten day trip to go from A to B.
At each town she must stay the night, and check into a motel at a cost that varies with each city.
It is also possible to go cross country, and drive diagonally in a north-easterly direction, but that is so tiring, that she will have to stay in the following town an extra day. It would therefore still take 10 days to go, say, from A to 23 to 17 to 11 to 5 to 6 to B, however, she might, by doing so, avoid a night in an expensive town.

Your problem is to find for her the cheapest route from A to B. The Text file (DATA31.txt for the first try and DATA32.txt for the second) consists of a list of 35 numbers, representing the price of a night at the local motel. Note that the price is always a 2-digit integer.
The first number is the price of the motel in town 1, followed by 2 etc. in order, including towns 7 and 29 (her departure and destination points)
Print the array of data and the shortest route as in the following example.

Sample Input

```
55 55 35 75 60 60 55
65 70 65 60 50 80 60
65 35 35 75 80 70 45
75 45 70 35 35 60 65
40 80 35 75 40 75 55
```

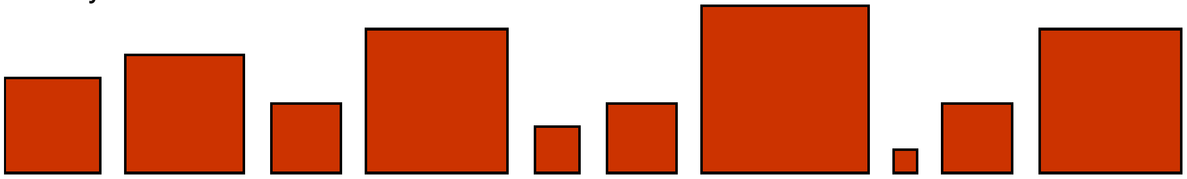


Sample Output

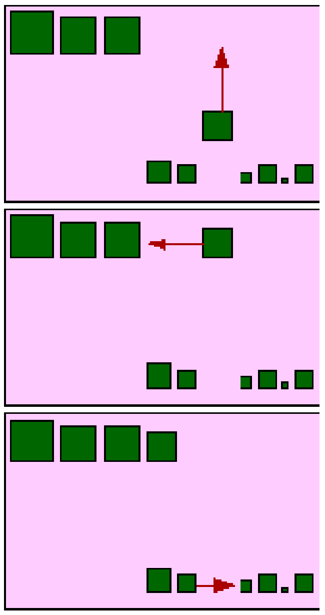
```
55    55    35    75    60    60    55
65    70    65    60    50    80    60
65    35    35    75    80    70    45
75    45    70    35    35    60    65
40    80    35    75    40    75    55

start in A, motel cost: $ 40
move East,  motel cost: $ 80
move East,  motel cost: $ 35
move NE,    motel cost: $ 70
move East,  motel cost: $ 35
move East,  motel cost: $ 60
move NE,    motel cost: $ 90
move North, motel cost: $ 60
move North, motel cost: $ 55
                          ---
            total cost: $525
```

Problem 4 – Sorting Squares

Imagine 10 squares, of random sizes, arranged along the top of a cleared screen. The sizes range from 1 unit to 12, and they are separated from their neighbours by exactly one unit.



Use animation to move the squares one at a time to the top of the window, and in order of size: largest to smallest.



The animation should follow the following paths:

1. First the targeted square should move up to the top, leaving a gap in the bottom row
2. The square then moves to the left
3. All the squares to the left of the gap should move right to close the gap

Continue this until all squares are in place.
There will only be one such sets of 10 squares.
Your program will read from an input file (DATA41.txt for the first try and DATA42.txt for the second) 10 numbers (between 1 and 12). The sum of this set of 10 numbers will never exceed 80 and be never less than 50. Each number will occupy a separate line.
Output should be on a cleared screen or window, and use most of the screen vertically and most of the screen horizontally. (For example, consider your screen to be 80 x 50 units, and adjust everything in proportion, relative the real number of pixels on your screen)

Marking scheme:
There are 5 conditions that must be met perfectly:
· 20 points for correctly listing the squares in their starting place, equally spaced.
· 20 points for correctly pulling each square in order out of their lineup
· 20 points for closing rank smoothly
· 20 points for lining them up correctly in their final space, equally spaced
· 20 points for the animation (leave no trails, clearly follow the designated path)
These marks will be assigned in order: if one condition is not followed correctly, subsequent conditions will not be marked.