ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

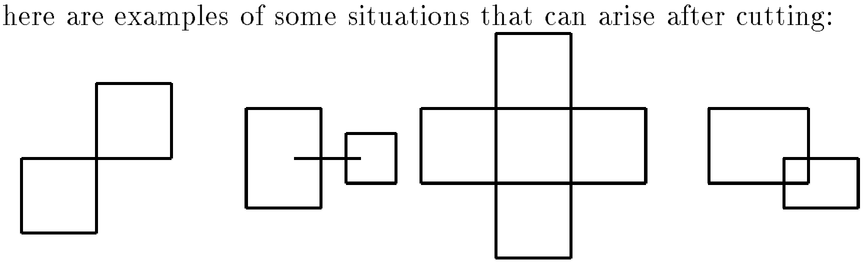
Problem A: Tin Cutter

Input file: `cutter.in`
Output file: `cutter.out`
Program file: `cutter.pas` or `cutter.cpp`

In a Tin Cutting factory there is a machine for cutting parts from tin plates. It has an extraordinarily sharp knife able to make horizontal or vertical segment cuts in the tin plates. Each cutting process consists of a sequence of such cuts. Each segment cut is given by its endpoints that are always located inside the tin plate. During the cutting process some parts of tin plate can fall out and so some holes in the plate can emerge.

Factory management needs to predict the number of holes in the plate at the end of the given sequence of cuts. Write a program that answers this question. Single segment cuts are not considered to be holes.

Here there are examples of some situations that can arise after cutting:



two holes two holes one hole one hole

Input

The input file consists of blocks of lines. Each block except the last describes one cutting process. In the first line of the block there is a number $N \leq 100$ indicating the number of segment cuts in the cutting process. These cuts are defined by the following N lines. The line defining one segment cut has the form X_1 Y_1 X_2 Y_2 where X_1, Y_1 and X_2, Y_2 are the co-ordinates of the end points of the segment cut. They are separated by one space. The co-ordinates are integers and always define horizontal or vertical segment (i.e. segment parallel with x or y axis). The last block consists of just one line containing 0.

Output

The output file contains the lines corresponding to the blocks in the input file. Each such line contains the number of holes that remain in the tin plate after the execution of the corresponding cuts. There is no line in the output file corresponding to the last "null" block of the input file.

Example

Input file:

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

Output file:

```
1
0
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

Problem B: FORCAL

Input file: `lexi.in`
Output file: `lexi.out`
Program file: `lexi.pas` or `lexi.cpp`

FORCAL is the programming language well known to programmers who are interested in compiler construction and especially to students attending Dr. C. Ompiler class. The FORCAL syntax is defined as follows:

- The only data type is integer.
- All identifiers are implicitly declared and are not longer than 32 characters. Identifiers are composed of letters, digits and underscores. At least one character of the identifier is not a digit.
- Literals are strings of at most 8 digits.
- Comments begin with `--` and end at the end of the line in which they start.
- Statement types are
  Assignment:
  $\langle$identifier$\rangle$:= $\langle$expression$\rangle$
  where expressions are constructed from identifiers, literals, operators $+$, $-$, and parentheses as follows:
  1) all identifiers and literals are expressions,
  2) if a, b are expressions then $a + b$, $a - b$, $+a$, $-a$, (a) are expressions.

  Input/Output:
  read(List of identifiers);
  write(List of expressions)
  (Items in the list are separated by comma)
- **begin**, **end**, **read**, and **write** are reserved words.
- Each statement is terminated by a semicolon.
- FORCAL is not case–sensitive, for example **BegIN** is the same keyword as **beGin**.

FORCAL tokens are defined to be: the identifiers or the literals or the symbols `+` `-` `(` `)` `:=` `;` `,` or the reserved words.

Notes:

- the assign operator is to be considered one FORCAL token,
- spaces, tabs, end-of-lines are allowed between the tokens,
- no part of any comment is a token,
- successive tokens that are either identifiers, literals or reserved words must be separated by a space or tab or end-of-line,
- no token is allowed to contain a space or a tab or end-of-line.

Help the students of Dr. C. Ompiler to write a program which reads lines of text an recognizes the FORCAL tokens in them.

Input

The input file consists of several blocks of lines. Each block contains lines of text and is terminated by one empty line.

Output

The output file consists of blocks corresponding to the blocks in the input file. In the lines of each block there are successively stored the FORCAL tokens recognized by the program (just one token on each line). Each token must be written on the output line in exactly the same form as it appears in the input text. If the program encounters a string that is neither a FORCAL token, nor comment, nor space, tab, end-of-line, it is to write the string `TOKEN ERROR` on a new line and continues by processing the next block in the input file. The program writes one empty line after each block of the output file.

Example

Input file:

```
A1:= A + (-B);

A123_A123 )
01.2 A B
C

:= A  beGIn

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Output file:

```
A1
:=
A
+
(
-
B
)
;

A123_A123
)
01
TOKEN ERROR

:=
A
beGIn

TOKEN ERROR
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

Problem C: L–system

Input file: `lsys.in`
Output file: `lsys.out`
Program file: `lsys.pas` or `lsys.cpp`

A D0L (deterministic Lindenmayer system without interaction) system consists of a finite set Σ of symbols (the alphabet), a finite set P of productions and a starting string ω. The productions in P are of the form $x \to u$, where $x \in \Sigma$ and $u \in \Sigma^+$ (u is called the right side of the production), Σ^+ is the set of all strings of symbols from Σ excluding the empty string. Such productions represent the transformation of the symbol x into the string u. For each symbol $x \in \Sigma$, P contains exactly one production of the form $x \to u$. Direct derivation from string u_1 to u_2 consists of replacing each occurrence of the symbol $x \in \Sigma$ in u_1 by the string on the right side of the production for that symbol. The language of the D0L system consists of all strings which can be derived from the starting string ω by a sequence of the direct derivations.

Suppose that the alphabet consists of two symbols `a` and `b`. So the set of productions includes two productions of the form $\texttt{a}\to u$, $\texttt{b}\to v$, where u and $v \in \{\texttt{a},\texttt{b}\}^+$, and the starting string $\omega \in \{\texttt{a},\texttt{b}\}^+$. Can you answer whether there exists a string in the language of the D0L system of the form xzy for a given string z? (x and y are some strings from Σ^*, Σ^* is the set of all strings of symbols from Σ, including the empty string.). Certainly you can. Write the program which will solve this problem.

Input

The input file of the program consists of several blocks of lines. Each block includes four lines. There are no empty lines between any successive two blocks. The first line of a block contains the right side of the production for the symbol `a`. The second one contains the right side of the production for the symbol `b` and the third one contains the starting string ω and the fourth line the given string z. The right sides of the productions, the given string z and the starting string ω are at most 15 characters long.

Output

For each block in the input file there is one line in the output file containing `YES` or `NO` according to the solution of the given problem.

Example

Input file:
```
aa
bb
ab
aaabb
a
b
ab
ba
```
Output file:
```
YES
NO
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

Problem D: Packets

Input file: `packets.in`
Output file: `packets.out`
Program file: `packets.pas` or `packets.cpp`

A factory produces products packed in square packets of the same height h and of the sizes 1×1, 2×2, 3×3, 4×4, 5×5, 6×6. These products are always delivered to customers in the square parcels of the same height h as the products have and of the size 6×6. Because of the expenses it is the interest of the factory as well as of the customer to minimize the number of parcels necessary to deliver the ordered products from the factory to the customer. A good program solving the problem of finding the minimal number of parcels necessary to deliver the given products according to an order would save a lot of money. You are asked to make such a program.

Input

The input file consists of several lines specifying orders. Each line specifies one order. Orders are described by six integers separated by one space representing successively the number of packets of individual size from the smallest size 1×1 to the biggest size 6×6. The end of the input file is indicated by the line containing six zeros.

Output

The output file contains one line for each line in the input file. This line contains the minimal number of parcels into which the order from the corresponding line of the input file can be packed. There is no line in the output file corresponding to the last "null" line of the input file.

Example

Input file:

```
0 0 4 0 0 1
7 5 1 0 0 0
0 0 0 0 0 0
```

Output file:

```
2
1
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

Problem E: Crosswords

Input file: `cross.in`
Output file: `cross.out`
Program file: `cross.pas` or `cross.cpp`

A crossword can be stored as a matrix $m \times n$ of zeros and ones. Zero represents white squares and one represents black squares. Some squares of the crossword are numbered and assigned to these numbers are the descriptions of the words that should be written either "across" or "down" into the crossword. A square is numbered if it is a white square and either (a) the square below it is white and there is no white square immediately above, or (b) there is no white square immediately to its left and the square to its right is white. Appropriate squares are numbered from left to right, from the top line to the bottom line.

From the matrix a crossword diagram can be drawn. In the diagram each square is represented by a box 4×6 characters. Black square is represented by

```
++++++
++++++
++++++
++++++
```

White squares are represented as follows (numbered and not numbered square):

```
++++++      ++++++
+nnn +      +    +
+    +      +    +
++++++      ++++++
```

where `nnn` is the number of the square.

The remaining characters of the box are spaces. If black squares are given at the edges, they should be removed from the diagram (see the example). Only use spaces as necessary filling characters. Don't use any unnecessary spaces at the end of the line.

Input

The input file consists of several blocks of lines each representing a crossword. Each block starts with the line containing two integers $m < 25$ and $n < 25$ separated by one space. In each of the next m lines there are n numbers 0 or 1, separated by one space. The last block will contain only one line with $m = n = 0$.

Output

The output file contains the corresponding crossword diagram for each except the last block. After each diagram there are two empty lines.

Example

Input file:

```
6 7
1 0 0 0 0 1 1
0 0 1 0 0 0 0
0 0 0 0 1 0 0
0 1 0 0 1 1 1
0 0 0 1 0 0 0
1 0 0 0 0 0 1
5 3
1 0 1
0 0 0
```

```
1 1 1
0 0 0
1 0 1
0 0
```

Output file:

```
     +++++++++++++++++++++
     +001 +    +002 +003 +
     +    +    +    +    +
++++++++++++++++++++++++++++++++++++
+004 +    ++++++005 +    +006 +007 +
+    +    ++++++    +    +    +    +
++++++++++++++++++++++++++++++++++++
+008 +    +009 +    +    +010 +    +
+    +    +    +    +    +    +    +
+++++++++++++++++++++    +++++++++++
+    ++++++011 +    +
+    ++++++    +    +
++++++++++++++++++++++++++++++++++++
+012 +013 +    ++++++014 +015 +    +
+    +    +    ++++++    +    +    +
++++++++++++++++++++++++++++++++++++
     +016 +    +    +    +    +
     +    +    +    +    +    +
     ++++++++++++++++++++++++++


     ++++++
     +001 +
     +    +
++++++++++++++++
+002 +    +    +
+    +    +    +
++++++++++++++++


++++++++++++++++
+003 +004 +    +
+    +    +    +
++++++++++++++++
     +    +
     +    +
     ++++++
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

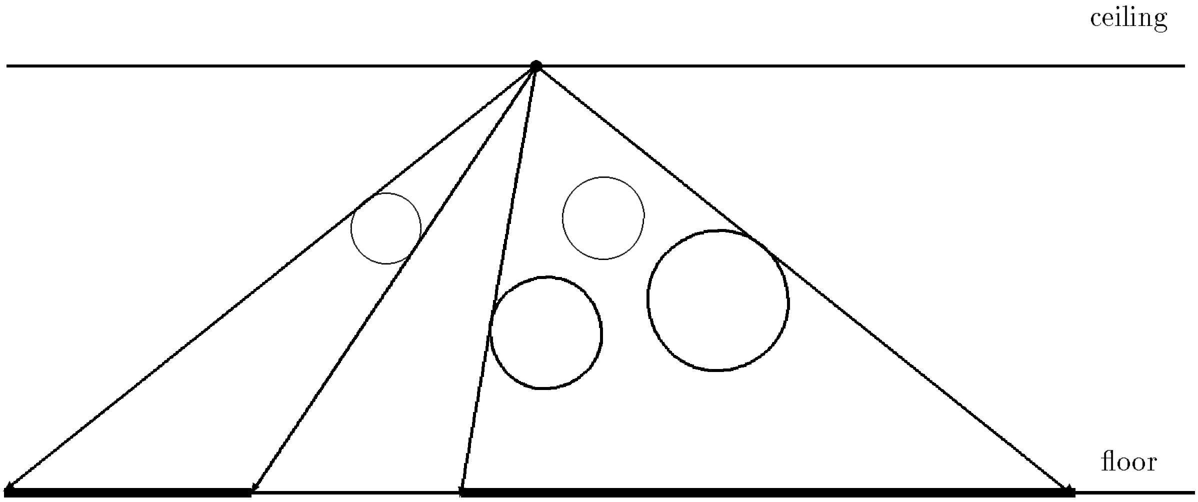
Central European Regional Contest

Problem F: Intervals

Input file: `interval.in`
Output file: `interval.out`
Program file: `interval.pas` or `interval.cpp`

In the ceiling in the basement of a newly open developers building a light source has been installed. Unfortunately, the material used to cover the floor is very sensitive to light. It turned out that its expected life time is decreasing dramatically. To avoid this, authorities have decided to protect light sensitive areas from strong light by covering them. The solution was not very easy because, as it is common, in the basement there are different pipelines under the ceiling and the authorities want to install the covers just on those parts of the floor that are not shielded from the light by pipes. To cope with the situation, the first decision was to simplify the real situation and, instead of solving the problem in 3D space, to construct a 2D model first.

Within this model, the x-axis has been aligned with the level of the floor. The light is considered to be a point light source with integer co-ordinates $[b_x, b_y]$. The pipes are represented by circles. The center of the circle i has the integer co-ordinates $[c_{i,x}, c_{i,y}]$ and an integer radius r_i. As pipes are made from solid material, circles cannot overlap. Pipes cannot reflect the light and the light cannot go through the pipes. You have to write a program which will determine the non-overlapping intervals on the x-axis where there is, due to the pipes, no light from the light source.



Input

The input file consists of blocks of lines, each of which except the last describes one situation in the basement. The first line of each block contains a positive integer number $N < 500$ expressing the number of pipes. The second line of the block contains two integers b_x and b_y separated by one space. Each of the next N lines of the block contains integers $c_{i,x}$, $c_{i,y}$ and r_i, where $c_{i,y} + r_i < b_y$. Integers in individual lines are separated by one space. The last block consists of one line containing $n = 0$.

Output

The output file consists of blocks of lines, corresponding to the blocks in the file (except the last one). One empty line must be put after each block in the output file. Each of the individual lines of the blocks in the output file will contain two real numbers, the endpoints of the interval where there is no light from the given point light source. The reals are exact to two decimal places and separated by one space. The intervals are sorted according to increasing x-coordinate.

Example

Input file:

```
6
300 450
70 50 30
120 20 20
270 40 10
250 85 20
220 30 30
380 100 100
1
300 300
300 150 90
1
300 300
390 150 90
0
```

Output file:

```
0.72 78.86
88.50 133.94
181.04 549.93

75.00 525.00

300.00 862.50
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

Problem G: Robot

Input file: `robot.in`
Output file: `robot.out`
Program file: `robot.pas` or `robot.cpp`

The Robot Moving Institute is using a robot in their local store to transport different items. Of course the robot should spend only the minimum time necessary when travelling from one place in the store to another. The robot can move only along a straight line (track). All tracks form a rectangular grid. Neighbouring tracks are one meter apart. The store is a rectangle $N \times M$ meters and it is entirely covered by this grid. The distance of the track closest to the side of the store is exactly one meter. The robot has a circular shape with diameter equal to 1.6 meter. The track goes through the center of the robot. The robot always faces north, south, west or east. The tracks are in the south–north and in the west–east directions. The robot can move only in the direction it faces. The direction in which it faces can be changed at each track crossing. Initially the robot stands at a track crossing. The obstacles in the store are formed from pieces occupying 1m $\times$ 1m on the ground. Each obstacle is within a 1×1 square formed by the tracks. The movement of the robot is controlled by two commands. These commands are `GO` and `TURN`.

The `GO` command has one integer parameter $n \in \{1, 2, 3\}$. After receiving this command the robot moves n meters in the direction it faces.

The `TURN` command has one parameter which is either `left` or `right`. After receiving this command the robot changes its orientation by $90°$ in the direction indicated by the parameter.

The execution of each command lasts one second.

Help researchers of RMI to write a program which will determine the minimal time in which the robot can move from a given starting point to a given destination.

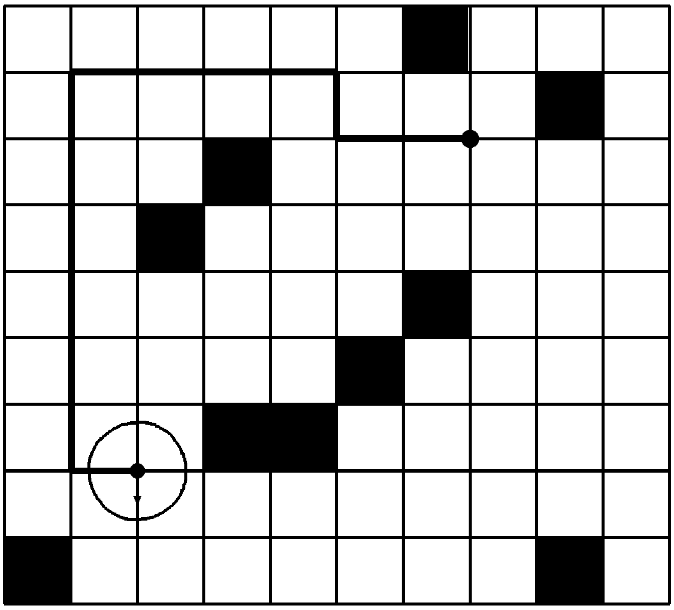
Input

The input file consists of blocks of lines. The first line of each block contains two integers $M \leq 50$ and $N \leq 50$ separated by one space. In each of the next M lines there are N numbers one or zero separated by one space. One represents obstacles and zero represents empty squares. (The tracks are between the squares.) The block is terminated by a line containing four positive integers B_1 B_2 E_1 E_2 each followed by one space and the word indicating the orientation of the robot at the starting point. B_1, B_2 are the coordinates of the square in the north–west corner of which the robot is placed (starting point). E_1, E_2 are the coordinates of square to the north–west corner of which the robot should move (destination point). The orientation of the robot when it has reached the destination point is not prescribed. We use (row, column)–type coordinates, i.e. the coordinates of the upper left (the most north-west) square in the store are 0,0 and the lower right (the most south-east) square are $M-1, N-1$. The orientation is given by the words `north` or `west` or `south` or `east`. The last block contains only one line with $N = 0$ and $M = 0$.

Output

The output file contains one line for each block except the last block in the input file. The lines are in the order corresponding to the blocks in the input file. The line contains minimal number of seconds in which the robot can reach the destination point from the starting point. If there does not exist any path from the starting point to the destination point the line will contain -1.

Example



Input file:

```
9 10
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 0
7 2 2 7 south
0 0
```

Output file:

```
12
```

ACM International Collegiate Programming Contest 96/97

Sponsored by Microsoft

Central European Regional Contest

Problem H: Network

Input file: `network.in`
Output file: `network.out`
Program file: `network.pas` or `network.cpp`

A Telephone Line Company (TLC) is establishing a new telephone cable network. They are connecting several places numbered by integers from 1 to N. No two places have the same number. The lines are bidirectional and always connect together two places and in each place the lines end in a telephone exchange. There is one telephone exchange in each place. From each place it is possible to reach through lines every other place, however it need not be a direct connection, it can go through several exchanges. From time to time the power supply fails at a place and then the exchange does not operate. The officials from TLC realized that in such a case it can happen that besides the fact that the place with the failure is unreachable, this can also cause that some other places cannot connect to each other. In such a case we will say the place (where the failure occured) is critical. Now the officials are trying to write a program for finding the number of all such critical places. Help them.

Input

The input file consists of several blocks of lines. Each block describes one network. In the first line of each block there is the number of places $N < 100$. Each of the next at most N lines contains the number of a place followed by the numbers of some places to which there is a direct line from this place. These at most N lines completely describe the network, i.e., each direct connection of two places in the network is contained at least in one row. All numbers in one line are separated by one space. Each block ends with a line containing just 0. The last block has only one line with $N = 0$;

Output

The output contains for each block except the last in the input file one line containing the number of critical places.

Example

Input file:

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

Output file:

```
1
2
```