Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

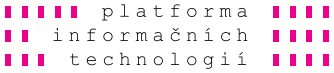
Welcome to CTU Open 2016 programming contest!

There are ten brand new original problems awaiting you. The problems were designed to challenge your programming wits, to test your skills and, sometimes, to put your perseverance on a trial. This was done on purpose. We enjoyed ourselves in preparing the problems and we wish you at least the same enjoyment with your own discoveries and breakthroughs. We hope that you are about to write fast, reliable and elegant solutions which you can be proud of.

Your programs can be written in C, C++, Java or Python programming languages. The choice is yours but you will be fully responsible for the correctness and efficiency of your solutions. We need the correct answer produced by your code in some appropriate time. Nothing else matters. You may choose any algorithm and any programming style.

All programs will read text from the standard input only. All inputs are terminated by the end of file mark. The results must be written to the standard output. You are not allowed to use any other files, communicate over network, or create processes. Input and output formats are described in problem statements and must be strictly followed. Numerical values on input line are separated by space. Do not print anything more than required. Each printed text line (including the last one) should be terminated by a newline character ("\n"), which is not considered to be a part of that line.

Good luck in the Czech Technical University Open Contest 2016!



This problem set consists of eleven sheets of paper (including this one) containing ten problems. Please make sure that you have the complete set.

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Aerial Archeology

`archeology.c`, `archeology.cpp`, `archeology.c11`, `Archeology.java`, `archeology.py`

Andrew is on a summer vacation job with a group of aerial archeologists. The group is internationally known for their advances in using nuclear imaging spectroscopy to investigate the underground remains of prehistoric cultures. Today, Andrew's job is to find a route for a helicopter which will carry the spectrometer over the area of archeological interest in the nearby lowlands. The spectrometer is a very sensitive and vulnerable device and the helicopter carrying it has to fly at constant speed in a perfectly straight line to minimize the measurement noise.

Hidden under the surface in the lowlands, there are more prehistoric settlements whose location and boundaries have been previously established by other techniques. All settlement boundaries are drawn on a special map which is at Andrew's disposal. The goal of the flight is to fly over as many settlements as possible and measure the soil composition in and around them. Thus, all Andrew has to do is to draw such straight line on the map that intersects the maximum number of settlements drawn there.

The shapes of the settlements are complicated and the settlements overlap, often chaotically. So it is not immediately obvious where to draw the line.

Input Specification

The input describes the shapes and the positions of settlements on the map. Each settlement is represented as a simple polygon (no two of its non-adjacent boundary segments touch or intersect each other). The polygons may overlap one another.

There are more test cases in the input. Each case starts with a line containing one positive integer N which specifies the number of polygons on the map. Then there is the description of N polygons. Each polygon description starts with one text line containing single integer M ($M \geq 3$) which denotes the number of vertices of the polygon. The next M lines specify the vertices of the polygon. Each of these lines specifies one vertex by its two coordinates x, y separated by space. The vertices are listed in the clockwise direction along the polygon boundary. All coordinates are integers with an absolute value at most 10 000. The total number of vertices of all polygons on the map does not exceed 1 000.

Output Specification

For each test case, print a single line with integer P denoting the maximum number of polygons on the map which can be intersected by a straight line. Note that only the intersections of the line with the *interior* of the polygons are considered.

Sample Input

```
3
4
0 0
0 1
1 1
1 0
4
1 2
1 3
2 3
2 2
5
2 1
2 2
9 2
10 3
10 1
```

Output for Sample Input

```
2
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Hot Air Ballooning

`balloon.c`, `balloon.cpp`, `balloon.c11`, `Balloon.java`, `balloon.py`

Dave is the director of the Summer school of hot air ballooning. Being a responsible director, he keeps a list of flights of each trainee in the school. After each flight, Dave appends a note to the lists of flights of each trainee participating in that particular flight. The note is very simple, it just indicates the type number of the balloon. In this way, each trainee flight history is characterized by a list of numbers.

At the end of the season, Dave wants to categorize the trainees according to their experience with different brands of balloons.

Two trainees belong to the same category if they have flown the same types of balloons. It does not matter how many times they have flown any particular balloon type, what does matter is the set of the balloon types they have flown and that has to be the same.

There are exactly nine types of balloons in Dave's school, and no trainee has flown more than nine times in a balloon, so Dave expresses each trainee list as an integer consisting of digits $1, 2, \ldots, 9$ and smaller than one billion. He thinks that this representation will help him to process the lists programmatically by a computer.

For example, the trainees characterized by integers 234423 and 342 belong to the same category, while the trainees characterized by integers 118821 and 1189821 belong to different categories.

Help Dave to calculate how many different categories of trainees attended the school this season.

Input Specification

There are more test cases. Each case starts with a line containing one integer N ($1 \leq N \leq 1\,000$) representing the number of trainees. Next, there are N lines, each line contains one integer representing the list of flights of one particular trainee.

Output Specification

For each test case, print a single line with one integer C specifying the number of different trainee categories in the school.

Sample Input

```
5
132
42
3312
43
24424
3
222
22
2
```

Output for Sample Input

```
3
1
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

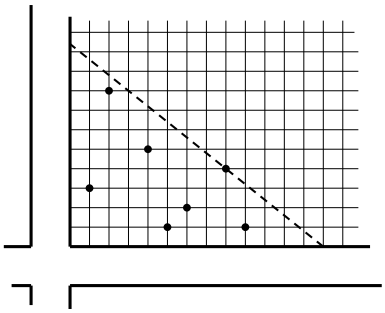
CTU Open Contest 2016

Cable Connection

`cable.c`, `cable.cpp`, `cable.c11`, `Cable.java`, `cable.py`

Two straight roads A and B perpendicular to each other start at the common crossing. Road A runs eastward and road B runs northward.

The roads are part of a large industrial system being built in the area and they should be connected by a special high frequency cable. Unfortunately, the connection cannot be built directly at the intersection. Additionally, there are some buildings standing in the corner of the field bordered by the roads. The buildings represent obstacles to the cable.



After some negotiations and considering various technical limitations, the analysts of the cable company constructed a set of critical points, determined by the obstacles. Then they suggested that the cable should connect the roads in a way that satisfies the following criteria:

- The cable should run in a straight line.
- The cable should not run between any critical point and the corner of the field at the roads crossing.
- The length of the cable should be the minimum possible.

Now, your task is to determine the length of the cable.

Input Specification

There are more test cases. Each case starts with a line containing one integer N ($1 \leq N \leq 10^6$) which specifies the number of critical points. Next, there are N lines representing the points, each line describes one of them. A point P is represented by two integers a, b ($1 \leq a, b \leq 10\,000$) separated by space. Integer a is the distance from P to road A and integer b is the distance from P to road B. All distances are in meters. You may suppose that the lengths of the roads and also the size of the field are not limited. All coordinate pairs (a, b) in one test case are unique.

Output Specification

For each test case, print a single line with one floating point number L denoting the minimum possible length of the cable expressed in meters. L should be printed with the maximum allowed error of 10^{-3}.

Sample Input

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

Output for Sample Input

```
16.648
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Rotating Display

`display.c`, `display.cpp`, `display.c11`, `Display.java`, `display.py`

Wendy is finishing her summer job in the lab testing a new 3D printed robot which is being taught to manipulate small objects.

To test the robot's abilities, a simple device called the display is used. It is a thin transparent square array of $N \times N$ square slots. Each slot contains a token, in the shape of an ASCII character (for better recognition), which is held in place in the slot by small magnets. The display can be rotated by 90° around the axis perpendicular to its surface or flipped by 180° around one of its four axes parallel to the display surface.

The robot simulates display rotations and flips by the following process. It removes the tokens from their slots and puts them back into different slots so that the contents of the display looks exactly as if the whole display was rotated or flipped. If, for achieving the desired effect, it is necessary to rotate or to flip particular tokens in their new positions the robot does it as well. The display remains stationary during the whole process.

For example, if the upper left corner of the display contains a token which looks like symbol "`<`" (less than) then after flipping the display around the vertical axis this token is moved to the upper right corner where it looks like the symbol "`>`" (greater than). Then, after left rotation the same token is moved back to the upper left corner where it looks like the symbol "`^`" (caret).

Wendy has programmed the robot to perform a long sequence of successive flips and rotations. To check the correctness of the robot's algorithms, she needs to know in advance how the display should look when the robot finishes its work.

Input Specification

We suppose that a token on the display can be shaped as any of the following so called symmetric characters: "`<`", "`>`", "`^`", "`v`", "`o`", "`x`", "`|`", "`-`", "`/`", "`\`". When a symmetric character is rotated or flipped it either remains the same or it becomes another symmetric character whose shape is the most similar to the rotated/flipped one.

There are more test cases. Each case starts with a line containing one integer N ($1 \leq N \leq 100$). Next, there are N lines representing the initial state of the display. Each line contains a string which consists of exactly N symmetric characters. Each character represents one slot on the display and the order of symbols in the input corresponds to the order of the tokens on the display. No slot on the display is empty. After N lines, there is a line with a command string specifying the flips and rotations that has to be carried out. A command string consists of command characters, each command character specifies one rotation or flip as follows: "`<`" (rotation left), "`>`" (rotation right), "`-`" (flip around horizontal axis), "`|`" (flip around vertical axis), "`\`" (flip around main diagonal), "`/`" (flip around anti-diagonal). Two successive command characters are separated by single space. The robot has to respect the order of commands in the command string. The number of commands is always positive and at most 10^6.

The decimal ASCII codes of the characters relevant to this problem are 45 (“-”), 47 (“/”), 60 (“<”), 62 (“>”), 92 (“\”), 94 (“^”), 111 (“o”), 118 (“v”), 120 (“x”), 124 (“|”).

Output Specification

For each test case, print N lines specifying the final position and orientation of the tokens on the display. The output format of the display representation is identical to the input format except for the size of the display which should not be printed.

Sample Input

```
3
o^-
/v|
vx^
< |
5
x>-o\
voooo
|ooo/
ooo/v
\o/vv
| \ |
```

Output for Sample Input

```
>-|
x<>
</o
<</o\
</ooo
/ooo|
oooo^
\o-<x
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Fence

`fence.c`, `fence.cpp`, `fence.c11`, `Fence.java`, `fence.py`

Joe and Marty invented a new game. The game is called Save the sheep and it is played on a rectangular grid. First, Joe draws a rectangle along the grid lines and then he fills some squares, not all squares, inside the rectangle with light gray color, those squares represent the sheep. Next, Marty fills one of the remaining squares inside the rectangle with black color, that square represents the wolf. Then they both separately strive to fulfill the objective of the game which is to draw the shortest possible fence around the sheep so that the wolf cannot reach them.

Specifically, the fence has to be drawn along the grid lines and it has to divide the grid into two areas. All sheep should be in the area enclosed by the fence while the wolf should remain in the area outside the fence. The fence must fit into the chosen rectangle, it may partially run along the border of the rectangle. ~~It must be a so called simple polygon, that is a polygon in which no two non-adjacent edges touch or intersect each other.~~

The player who draws the shortest fence wins. Sometimes, it is not possible to draw the desired fence and in such a case both players lose the game. Your task is to write a program that wins the game whenever it is possible.

Input Specification

There are more test cases. Each case starts with a line containing two integers M, N ($1 \leq M, N \leq 1000$) separated by space which represent the height and the width of the rectangle.

Next, there are M lines representing the contents of the rectangle. Each line specifies one row of the rectangle and it contains a string of length N. Each character in the string represents one grid square and it is either capital letter "`X`", capital letter "`O`" or symbol "`.`" (dot). Letter "`X`" represents the wolf, letter "`O`" represents the sheep and the dot "`.`" represents an unoccupied square of the grid. Each animal occupies exactly one square in the grid and there is exactly one wolf and at least one sheep in the rectangle.

Output Specification

For each test case, print a single line with one integer denoting the length of the shortest fence that separates the sheep from the wolf according to the rules of the game. We suppose that each square side in the grid has unit length. If it is not possible to draw the fence print "`-1`".

Sample Input

```
5 5
.....
..OOO
..OXO
O...O
..OOO
3 5
..O..
.OXO.
..O..
2 3
.O.
OXO
1 3
OXO
```

Output for Sample Input

```
26
-1
12
-1
```

Erratum

Due to an incorrect assumption, the original problem statement (as used in the 2016 CTU Open Contest) required the fence to form a *simple polygon*. With this requirement, the problem turned out to be much harder than expected. The test data and both authors' solutions correspond to a variant of the problem under the following conditions:

- The fence may go across the same *point* twice, thus not being a simple polygon.
- It is still forbidden to use the same grid *line* more than once.
- The wolf must not be trapped by the fence. In other words, grid squares outside the rectangle must be accessible to the wolf without crossing the fence.

The authors apologize for this error and express their gratitude to all teams participating in the 2016 CTU Open Contest for being so clever that no one even attempted to submit the incorrect solution (that would be accepted). This helped a lot to keep the results of the competition valid and fair.

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Tree Stands

`huntsmen.c`, `huntsmen.cpp`, `huntsmen.c11`, `Huntsmen.java`, `huntsmen.py`

Tree stands are elevated wooden platforms attached to trees. Typically, huntsmen use tree stands to watch or to shoot their prey.

In our county, huntsmen have built a remarkable system of tree stands. The tree stands are connected by narrow straight paths which form a kind of maze on the hunting grounds. The builders wanted to minimize the impact on the environment and so they built the minimum possible number of paths which ensure that there is a connection between any two tree stands. Also, a tree stand is visible from another stand if and only if the two are connected by a path.

A group of local huntsmen wants to find out which particular tree stands will serve the best their hunting interests. Each day they climb a different set of tree stands and watch the wildlife.

There are a few more important circumstances to consider:

- Security rules dictate that any occupied tree stand must be visible from at least one other occupied tree stand so that in case of an emergency the huntsman in the neighbour tree stand can come to help the colleague.
- A tree stand is always occupied by at most one huntsman.
- It does not matter which huntsman is in which tree stand. It only matters which tree stands are occupied and which are not.
- The size of the group does not change.

How many days will the group spend in the tree stands before they investigate all possible choices of tree stands available to them?

Input Specification

There are more test cases. Each case starts with a line containing two integers N and K ($2 \leq K \leq N \leq 200$) separated by space. N is the number of tree stands, K is the size of the group of huntsmen. The tree stands are labeled $1, 2, 3, \ldots, N$.

Next, there are $N - 1$ lines, each line specifies one path between two tree stands. The line contains the labels of the stands separated by a space. The order of the labels on a line and the order of the paths in the input is arbitrary.

Output Specification

For each test case, print on a separate line the number of days which the group will spend in the tree stands. Express the result modulo $1\,000\,000\,007$.

Sample Input

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

Output for Sample Input

```
3
3
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Orchard Division

`orchard.c`, `orchard.cpp`, `orchard.c11`, `Orchard.java`, `orchard.py`

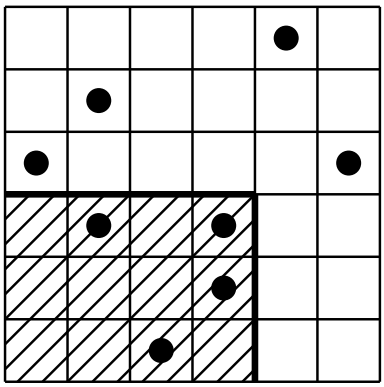
Uncle Oliver is going to sell a significant part of his famous dwarf plum tree orchard. He is going to divide the orchard into two parts, sell the first one and keep the other one.

The trees were originally planted in regular rows and columns forming a rectangular grid with the same number of rows and columns. As years went by, Oliver removed many trees which were weak or plagued by bugs so nowadays there is also a lot of free squares unoccupied by any tree.

Oliver has decided that he will keep exactly half of all the trees in the orchard. Moreover, he has few additional demands which, in his opinion, will ensure easy maintenance of his part in the future.

- The part Oliver is going to keep should be in the shape of a rectangle.
- A least one corner of the rectangle should coincide with a corner of the orchard.
- The rectangle area should be as small as possible.

Originally, each tree was planted in the center of an imaginary square whose area was exactly one square meter. Thus, the position of each tree can be described by the coordinates of the square on which it is standing. The dividing fence between the two parts of the orchard will run along the borders of the squares.



Input Specification

There are more test cases. Each case starts with a line containing two integers M ($1 \leq M \leq 10^9$) and N ($1 \leq N \leq 10^6$) separated by space. The orchard side length in meters is expressed by M and the number of trees in the orchard is expressed by N. Next, there are N lines, each line specifies x and y coordinates of one tree in the orchard. The coordinates are separated by space. For simplicity reasons, we assume that the coordinates are zero based, so the coordinates of the squares in the corners of the orchard are $(0, 0), (0, M-1), (M-1, M-1), (M-1, 0)$. All coordinate pairs (x, y) in one test case are unique.

Output Specification

For each test case, print a single line with one whole number A denoting the minimum possible area in square meters of uncle Oliver's part of the orchard. If it is not possible to divide the orchard according to Oliver's demands print "`-1`". Note that the output value might not fit into 32-bit integer type.

Sample Input

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

Output for Sample Input

```
12
-1
1
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

It's Raining, Man

`raining.c`, `raining.cpp`, `raining.c11`, `Raining.java`, `raining.py`

Walter is spending holidays at the farm of his great-grandfather. It is raining and raining and raining. Walter sits in the attic of an aged barn where he has found a pack of old poker cards among piles of various dusty junk. The cards look quite antiquated and interesting. He starts to lay down the cards one by one, side by side, on the floor when he suddenly notices that there appears to be some kind of order in their sequence. Many pairs of successive cards are either of the same rank or of the same suit. "This might be a nice little puzzle," says Walter to himself. "I wonder if I can rearrange the sequence so that each two consecutive cards share either the rank or the suit. But wait, the pack looks to be incomplete, that might severely limit the arrangement possibilities, hmm..."

Help Walter determine whether his puzzle is solvable.

Input Specification

There are more test cases. Each test case consists of a single line on which all cards in the pack are listed. The list starts with one integer L ($1 \leq L \leq 52$), denoting the number of cards in the pack, followed by a space and L card descriptions. Each card is described by a two character string. The first character denotes the rank of the card ("`A`"=Ace, "`2`"–"`9`", "`X`"=10, "`J`"=Jack, "`Q`"=Queen, "`K`"=King) and the second character denotes the suit of the card ("`C`"=Clubs, "`D`"=Diamonds, "`H`"=Hearts, "`S`"=Spades). The successive card descriptions are separated by one space.

Output Specification

For each test case, print a single line with the string "`YES`" if the puzzle is solvable or a line with the string "`NO`" if the puzzle is not solvable.

Sample Input

```
8 2C 2D 2H 2S XC JS QS KS
4 5C 4H AS 9D
```

Output for Sample Input

```
YES
NO
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Suspicious Samples

`samples.c`, `samples.cpp`, `samples.c11`, `Samples.java`, `samples.py`

Fatima is a researcher. She currently studies water circulation in her country river basins. She collects data samples from various meteorological stations that measure diverse climate-related values. Fatima then searches for interesting patterns in those samples. She uses a program which reads incoming samples' data in real time and outputs those samples which are interesting or suspicious in some way. The decision whether a sample is "interesting" or "suspicious" is based on a fixed set of conditions, such as "the value is greater than the average of the last two hours" or "the value is lower than anything else in the last five minutes" which are easy to program into a computer.

Today, Fatima is in doubt about her yesterday's results and she came to see you, an experienced programmer. She thinks that her program did not evaluate the data correctly and she asks you to help her verify its results. In particular, she brings the complete sequence of samples and describes the set of conditions to you. Your program has to read the samples and produce the output according to the conditions. Fatima will then compare the output of your program to the output of her program and decide what has to be done next.

Input Specification

There are more test cases. The first line of each test case contains one integer N ($1 \leq N \leq 10^5$), giving the number of samples. Then there are N lines, each describing one sample. The line contains two integers T_i and V_i ($1 \leq T_i \leq 10^9$, $1 \leq V_i \leq 10^4$), meaning that the sample value V_i was acquired in time T_i. The times are given in seconds elapsed since some fixed moment in the past and they form a strictly increasing sequence ($\forall i, k\ 1 \leq i < k \leq N : T_i < T_k$).

The next line of the input contains one integer C ($1 \leq C \leq 10$), the number of conditions to evaluate. Each of the following C lines specifies one condition C_j. The line contains three tokens separated with a space:

- A relation operator R_j, which is either "`gt`" (greater than) or "`lt`" (less than).
- An aggregate function F_j, one of the "`min`" (minimum), "`max`" (maximum), or "`avg`" (average).
- An integer number L_j specifying the length of the time interval to be concerned, in seconds.

In general, a condition applied to a sample value V_i checks how V_i is related to some aggregate feature of the samples which were acquired before V_i. The function F_j specifies exactly that feature.

To be more specific, let S_{ij} be the set of all samples which were acquired before V_i but no more than L_j seconds earlier. The sample value V_i satisfies the condition C_j if and only if the relation $V_i R_j F_j(S_{ij})$ holds. For example, the sample value 800 in conjunction with "`lt min 300`" can be read as "is 800 less than the minimum sample value acquired in the previous 5 minutes before this 800 was obtained?". Note that sample V_i is not an element of S_{ij}.

Output Specification

For each condition, print one integer: the number of samples whose values satisfy that particular condition. If there are no samples in the time interval specified by the condition, the condition is never considered satisfied.

Sample Input

```
10
60 30
120 28
180 35
240 34
300 40
360 31
420 28
480 2
540 42
600 30
2
gt avg 7200
lt min 300
```

Output for Sample Input

```
4
2
```

Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2016

Colorful Tribune

`tribune.c`, `tribune.cpp`, `tribune.c11`, `Tribune.java`, `tribune.py`

You are visiting your friends on a summer holiday sports camp. When you arrive there is a volleyball tournament going on. Various clubs from the villages in the camp neighborhood are taking part in the tournament. Each club is recognized by a color which is different from the colors of all other clubs and also the fans of each particular club are wearing t-shirts with the color of their club.

Amazingly, the tribune on which the fans are sitting is suited very well for the tournament: The number of seat rows and also the number of seat columns (perpendicular to the seat rows) is equal to the number of the clubs participating in the tournament. Moreover, quite unbelievably, the number of fans of each club is also equal to the number of clubs.

The last game of the tournament is to begin shortly. To make the display on the tribune more attractive, the fans decided to arrange themselves in such way that each fan will occupy exactly one seat and each row and each column will be occupied by fans of all clubs. After some scramble, they manage to achieve this configuration and the game is about to start. Suddenly, someone in the crowd around you points to the tribune and shouts "Hey, John is not wearing his club's color!". You do not know who John is and therefore you do not know where is he sitting or what color he should be wearing. Nevertheless, if John is the only person on the tribune who failed to be dressed in proper color then it is possible, just by careful observation of the tribune, to find him and determine the color of his club.

Input Specification

In this problem, we denote the colors of the clubs by capital letters "`A`", "`B`", ..., "`Z`". There are more test cases. Each case starts with a line containing one integer N ($3 \leq N \leq 26$) which is the number of teams taking part in the tournament. Next, there are N lines describing the colors worn by the fans on the tribune. Each line corresponds to one row of sets and it contains one string of length N. Each character in the string represents the color which the corresponding fan is wearing. The order of seats on the tribune is the same as the order of characters in the input.

Output Specification

For each test case, print a single line with two integers R, C, and a character V. The values of R and C specify the row and the column on the tribune where John is sitting and the letter V specifies the color of his club. Rows and columns are numbered $1, 2, \ldots, N$. The successive values on the output line should be separated by one space.

Sample Input

```
6
OEYCDK
EYOKCD
KDCEOY
CKHOYE
YOEDKC
DCKYEO
3
IWL
GIW
WLI
```

Output for Sample Input

```
4 3 D
2 1 L
```