Problem 1: Corridor

Pat and Mat are carrying a very special package through a peculiar corridor. The package is very long and thin and, as its content is very delicate (don't ask), it cannot be rotated—it can be only shifted. Can Pat and Mat get the package to the other side of the corridor?

To visualize their predicament, let's look at the situation from above. The corridor is formed by its left and right walls that each form a non-self-intersecting polygonal curve (a polygonal curve is a sequence of line segments where, for two consecutive line segments, the ending point of the first segment is the same as the starting point of the second segment). Both walls start on the x-axis and when traversing their curves, the y coordinate always increases. The endpoints for the two walls share the same y-coordinate. Of course, the left and right wall never cross. Pat's and Mat's package is parallel to the x-axis—can they get it through the corridor while keeping the package parallel to the x-axis?

Input specification:

The first line contains k, the number of corridors. Each corridor is described on several lines. The first line is empty. The second line contains three positive integers m, n, and d, where m is the number of points that define the left wall, n is the number of points that define the right wall, and d is the length of the package. The third line contains $2m$ integers $a_{x,1}, a_{y,1}, a_{x,2}, a_{y,2}, \ldots, a_{x,m}, a_{y,m}$, the coordinates of the points on the left wall, where $0 = a_{y,1} < a_{y,2} < \cdots < a_{y,m}$. The fourth line contains $2n$ integers $b_{x,1}, b_{y,1}, b_{x,2}, b_{y,2}, \ldots, b_{x,n}, b_{y,n}$, the coordinates of the points on the right wall, where $0 = b_{y,1} < b_{y,2} < \cdots < b_{y,n} = a_{y,m}$. All input integers are, in absolute value, smaller than 10,000,000, $a_{x,1} < b_{x,1}$, and $m, n \geq 2$.

Output specification:

The output contains k lines. The i-th line corresponds to the i-th corridor. It contains `YES` if it possible to get the package through the corridor, and `NO` otherwise.

Sample input:

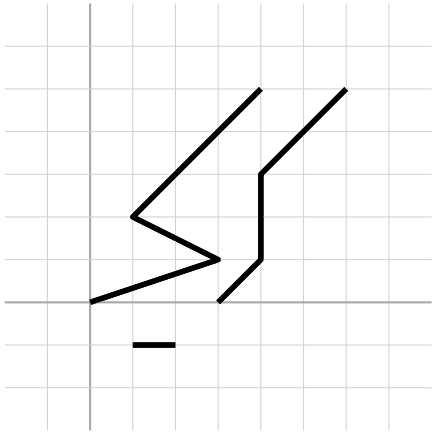
```
2

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

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

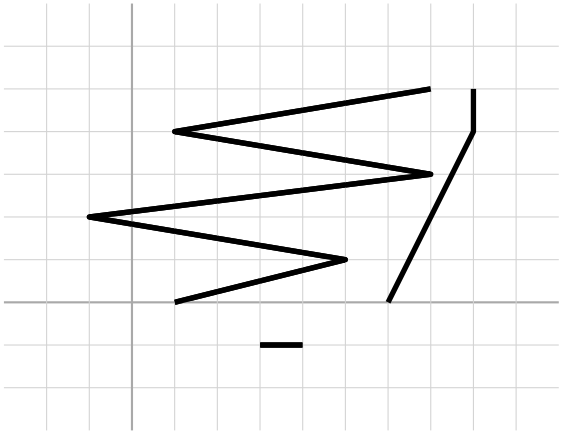
Sample output:

```
YES
NO
```

Explanation:

The two sample inputs are depicted below—sample input 1 on the left, sample input 2 on the right.





Problem 2: DVDs

Dezider is very unhappy as he just discovered that his DVDs got unsorted. Dezider is (occasionally) very organized and during one of his get-organized spells he numbered the DVDs from 1 to n. He keeps the DVDs in a tall stack and he wants to have them sorted in increasing order by their number, with 1 at the bottom and n at the top of the stack. The trouble is that his space allows him to perform only one type of sorting operation: take a DVD, pull it out of the stack while the DVDs above it fall down by one position, then place it at the top of the stack. What is the smallest number of such operations he needs to do to sort the DVD stack?

Input specification:

The first line contains k, the number of input instances. Each input instance is described on two lines. The first line contains $n \leq 10{,}000{,}000$. The second line lists the DVDs in the initial order on the stack, from the bottom to the top.

Output specification:

The output contains k lines. The i-th line corresponds to the i-th input. It contains the smallest number of operations needed to sort the stack.

Sample input:

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

Sample output:

```
1
2
```

Explanation:

In the first sample input it suffices to take DVD 4 and move it to the top of the stack; then the stack is sorted. In the second sample input one can first move DVD 4 and then DVD 5 to get a sorted stack.

Problem 3: Maze

Filip (and his dad) started experimenting with robotics. They built a robot that moves through a maze drawn on a rectangular graph paper with some grid squares colored black. The robot's color sensor prevents it from going on or through the black regions—they act like walls. The graph paper is on a huge black board, guaranteeing that the robot stays on the paper. As for the robot's moves, it "jumps" from one grid square to one of the four adjacent grid squares (this required some very fine precision and they are justly proud of their work). There is only one problem: the robot stubbornly refuses to jump more than twice in a row in the same direction. It also refuses to jump backwards to the location it just came from (though it can revisit the same location later).

After watching the robot skip happily through the maze, Filip decided that it needed a destination. He colored one of the grid squares red and with dad's help reprogrammed the robot to start singing once it finds the red square.

Sometimes Filip needs to wait for the singing for a very long time since the robot usually meanders through the maze. In such cases, Filip wonders how long would it take to move along the shortest route. Naturally, you want to help him: compute the smallest number of steps the robot needs to make to get to the red square.

Input specification:

The first line contains k, the number of mazes. Each maze is described on several lines. The first line is empty. The second line contains two integers a and b, the dimensions of the graph paper. Then a lines follow, each with a string of length b. The strings contain only symbols '.' corresponding to an uncolored grid square, 'B' corresponding to a black square, 'R' corresponding to the robot, or 'D' corresponding to the destination (red square). You may assume that $a, b \leq 1{,}000$ and that there is exactly one robot (standing on an uncolored square), and exactly one red square.

Output specification:

The output contains k lines. The i-th line corresponds to the i-th maze. It contains `-1` if there is no route through the maze that leads to the red square. Otherwise, it contains the smallest number of steps needed to get to the red square.

Sample input:

```
4

5 6
....D.
.R.B..
...B..
......
......
```

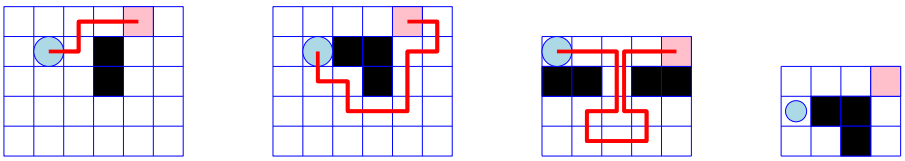
```
5 6
....D.
.RBB..
...B..
......
......

4 5
R...D
BB.BB
.....
.....

3 4
...D.
RBB.
..B.
```

Sample output:

```
4
10
14
-1
```

Explanation:

Shortest routes for the first three sample inputs are shown below. There is no route for the fourth sample input, shown on the right.



Problem 4: Palindrome

Mr. F. just loves palindromes (a palindrome is a word that reads the same forwards and backwards). He got hold of his mom's solved crosswords and now he is looking for palindromes in them. The crosswords are somewhat unusual: each is an $n \times n$ grid with no black squares—every location contains a letter. Mr. F. considers only sequences of letters that can be read by going right and down, that is, the next letter in the sequence is immediately to the right or below the current letter. Help him find the longest palindrome.

Input specification:

The first line contains k, the number of crosswords. Then k crossword descriptions follow. Each crossword description starts with a number $n \in \{1, \ldots, 100\}$. Then an $n \times n$ matrix of uppercase letters follows (there are n letters in each row; the letters are not separated by spaces and the row ends with a new line).

Output specification:

The output contains k lines, one for each crossword. The i-th line contains four items. The first item is one of the longest palindromes that occur in the i-th crossword. The second and third item specify the row and column coordinates of the first letter in the palindrome, with top left corner at coordinates 1,1. The fourth item is a string consisting of only letters R, D, and S that indicate how to move through the crossword to get the palindrome: starting at the specified location, move to the right if the current letter is R or down if the letter is D; stop at letter S, the last letter of the string.

Sample input:

```
2
3
ABA
BAB
ABA
4
ABCD
BCDA
CDAB
DABC
```

Sample output:

```
ABABA 1 1 RRDDS
B 1 2 S
```

Explanation:

A longest palindrome in the first sample input is ABABA. There are several occurrences of this palindrome, the sample output lists the one that starts at location 1,1 and then goes right-right-down-down.

For the second sample input there are only palindromes of length 1. One such palindrome is B, that can be found at location 1,2.

Problem 5: Pegs

Little Zorro and Worro are playing an interesting game on an $n \times n$ square grid. Each game is really short: each player makes just one move. The game is played with very very thin pegs. The pegs come in two colors: black and white. Zorro goes first: he puts the pegs at the vertices of the grid. Then Worro selects a subset of the pegs by drawing a convex polygon—whatever is inside or on the boundary of the polygon is in his set. (The pegs are so tiny that they act as points in the plane.) For each black peg selected Worro gets \$1. For each white peg selected Worro has to pay \$1. Naturally, Worro wants to get as much money as possible. Now he is worried that he might not choose the optimal set. Please, help him.

Input specification:

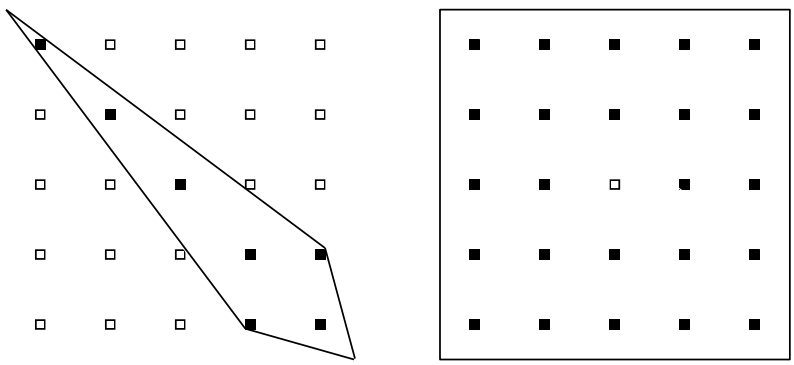
The first line contains k, the number of games. Then k game descriptions follow. Each game description starts with a number $n \in \{1, \ldots, 20\}$. Then an $n \times n$ zero-one matrix follows—ones represent black pegs, zeros represent white pegs.

Output specification:

The output contains k lines, one for each game. The i-th line contains the largest amount of money Worro can win in the i-th game.

Sample input:

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



Sample output:

```
7
23
```

Explanation:

Examples of convex polygons that achieve the values given in the output are above. In the first example Worro gets \$7, in the second he gets \$24-\$1=\$23.

Problem 6: Pipes

Kazimír got an important job at the plumbing department. He needs to connect certain pairs of locations by pipes. However, as the department's budget is very low, they have only two types of pipe parts: straight and "elbow" (L) shaped ones, all of the same size. Moreover, each pipe part is embedded into a square tile and the department provides a grid board onto which one can easily attach these tiles. The tiles can be rotated. The advantage of this design is that when two adjacent tiles' pipes align properly, there is no leaking. However, not having the option to go into the third dimension somewhat limits Kazimír's connection options. Help him figure out whether he can connect the pairs of locations!

The grid board is of dimension $m \times n$. There are ℓ pairs of locations that need to be connected. Every location is along a side of the board, in the middle of a side of a grid square. Once Kazimír places a pipe tile on a grid square, he is not allowed to place another tile on the same square.

Input specification:

The first line contains k, the number of input instances. Each input instance is described on several lines. The first line contains three positive integers m, n, and ℓ. The second line contains 2ℓ numbers $s_1, t_1, s_2, t_2, \ldots, s_\ell, t_\ell$ that describe the pairs of locations: Kazimír needs to connect location s_i to t_i for every $i \in \{1, \ldots, \ell\}$. The locations are described by numbers between 1 and $2(m + n)$, with 1 being the left side of the square in the top-left corner, then 2 is the left side of the square immediately below, 3 below that and so on, going in counterclockwise order around the boundary of the board; $2(m+n)$ is the top side of the top-left square. You may assume that $1 \leq m \times n \leq 10{,}000$ and that there are no duplicate locations – all the s_i and t_j numbers are distinct.

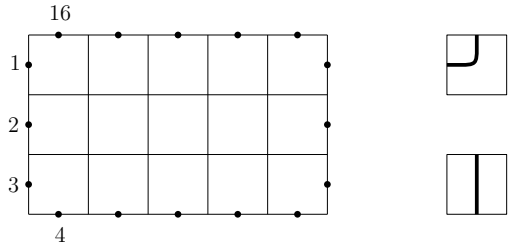


Figure 1: Numbering of locations for an $m \times n = 3 \times 5$ board. The types of tiles are shown on the right.

Output specification:

The output contains k lines. The i-th line corresponds to the i-th input. It contains `YES` if it is possible to connect all the pairs of locations and `NO` otherwise.

Sample input:

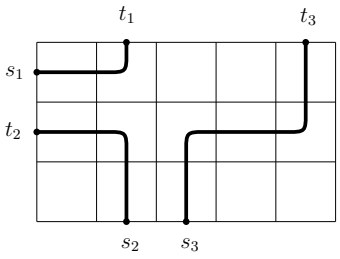
```
3
3 5 3
1 15 5 2 6 12
3 5 3
1 12 5 2 6 15
3 3 4
1 11 4 2 6 9 5 10
```

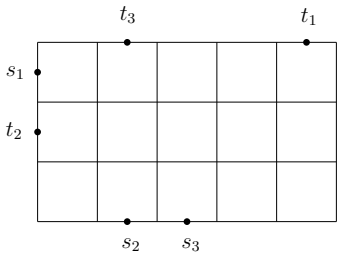
Sample output:

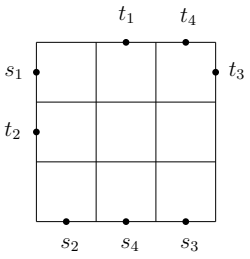
```
YES
NO
NO
```

Explanation:

The three sample inputs are depicted below—sample input 1 on the left, with possible pipe connections between the locations; sample inputs 2 and 3 are in the middle and on the right, respectively.

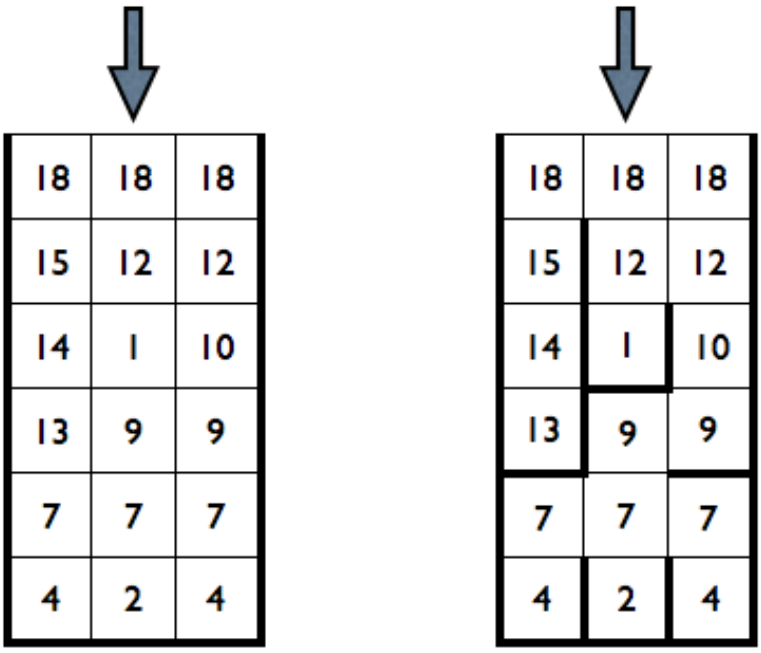






Problem 7: The Innovative Water Clock

Simon has been reading about water clocks in school - water pours into a vessel at a constant rate, so that the amount of water in the vessel indicates the amount of time that has passed. He has decided to make such a clock, but thinks that the traditional method of a simple cylinder with lines up the side to note the passage of time is too boring. Instead, he would like his containers to fill up in interesting patterns. His clocks will be rectangular (when viewed from the side) and with some internal walls so that different areas (grid cells) of the clock will fill up at different times. He will give you a picture of the side view of the clock he would like to build. In his clocks, water will be poured in at the rate of one grid cell per minute, and each number given represents which minute that particular square will become completely full. Here is an example:



On the left is Simon's specification. On the right, we can see the necessary walls. After 1 minute, the cell with the 1 will be full. After 1 more minute (2 minutes total), the water overflowing the "1" cell will fill up the cell marked "2". The two cells marked "4" will fill up at the same time, so they both fill after 2 more minutes (4 minutes total), and so on. Of course, Simon doesn't want to have to build more walls than necessary, so you must find the smallest set of walls that satisfies his specification.

Input specification:

The first line contains k, the number of input instances. The first line of each input instance contains three numbers: m, the number of rows of the clock, n, the number of columns of the clock, and c, the column into which the water source will pour ($c = 1$ represents the left-most column), with $m \times n < 40$. This is followed by m lines, each containing n integers representing the required fill times for the corresponding cell of the clock.

Output specification:

The output will consist of the solution for each input instance, in order, separated by a single blank line. For each input instance, the output will consist of $2m$ lines. Each odd line represents the cells of the clock (represented with `'o'`) and, if present, the vertical wall segments (represented with `'|'`) including those on the outside of the clock. If no vertical wall is present in a location, a space should be printed instead. Each even line represents the boundary between two rows of cells. This should consist of spaces except where a horizontal wall segment is present, which should appear as a hyphen (`'-'`).

Sample input:

```
1
6 3 2
18 18 18
15 12 12
14 1 10
13 9 9
7 7 7
4 2 4
```

Sample output:

```
|o o o|

|o|o o|

|o|o|o|
   -
|o|o o|
 -   -
|o o o|

|o|o|o|
 - - -
```

Explanation:

This shows the input and output for the example given above in pictures.

Problem 8: WordSpin

Žofka invented a new word puzzle. She gives you two strings s_1 and s_2 of the same length. You need to modify s_1 into s_2 as quickly as possible. The trick is that you are allowed to modify the strings only using the following types of moves: (1) shift forward where you choose a substring of one of the strings and shift each of its letters by 1 forward in the alphabet, or (2) shift backward where you shift each letter in a substring backward in the alphabet. The first move is not allowed if the substring contains the letter `z` while the second move is not allowed if the subtring contains `a`. What is the smallest number of moves you need to modify s_1 into s_2?

Input specification:

The first line contains k, the number of word puzzles. Each word puzzle is described on a single line that contains the strings s_1 and s_2 separated by space. The strings contain only lower case letters. You may also assume that the length of each string is at most 10,000,000.

Output specification:

The output contains k lines. The i-th line corresponds to the i-th word puzzle. It contains the smallest number of moves needed to modify s_1 into s_2.

Sample input:

```
2
hello teams
aacccaaaa bbbbbbbbb
```

Sample output:

```
27
3
```

Explanation:

The first sample input can be modified in the following way. First shift `lo` forward, getting `helmp`. Then shift `h` forward 12 times, getting `telmp`. Then shift `l` 11 times backward to get `teamp` and then shift `p` forward three times to get `teams`. Total number of moves is 1+12+11+3=27.

The second sample input can be modified as follows. First shift the entire string forward, getting `bbdddbbbb`. Then shift `ddd` backward twice to get `bbbbbbbbb`. This requires 1+2=3 moves.