French-Australian Regional Informatics Olympiad

9th March, 2007

Duration: 4 hours

4 questions

All questions should be attempted

Problem 1
Rescue Robot

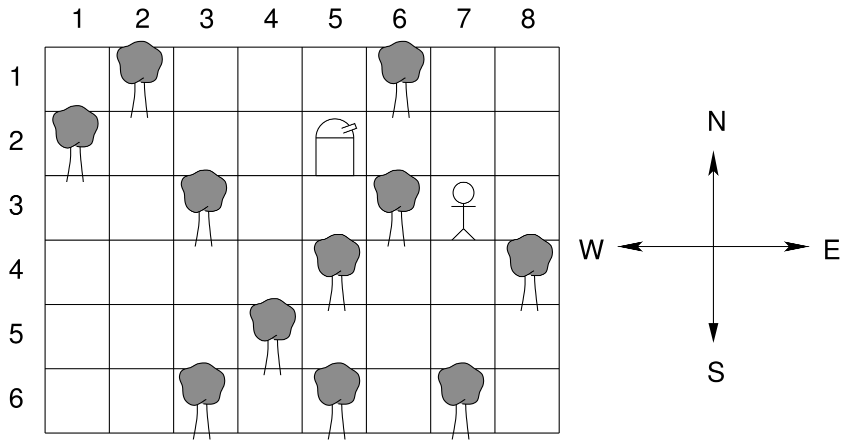
Time and Memory Limits: 1 second, 32 Mb

You are lost in a vast forest with nothing but monkeys and budgies for company. For days you have been trying to find your way out, but no matter how hard you try, you seem to find yourself back where you started. Resigned, you decide to stay put and wait for a rescue party.

Days and nights pass, until finally a search robot stumbles upon you and brings you food, water and a powerpoint to recharge your laptop! It's not the rescue party you were hoping for, but you're saved. Well, nearly.

The robot is unable to tell you how to get back to the rescue party. All it can do is print out a map of the forest, the location at which it found you, and the steps that it took from the rescue party in order to find you. Unfortunately there is a bug in the robot's program — if the robot tried to move into a tree or step outside the forest, it would still record this step, but it would not actually move.

For example, imagine the forest as the grid below, with some of the squares containing trees. If the robot started at grid square $(5, 2)$ and took the steps `E`, `E`, `E`, `S`, `S`, `W`, it would finish in square $(7, 3)$ where you are waiting (note that it does not actually move during the second `S` step because there is a tree in the way).



On the other hand, suppose the robot started at square $(7, 1)$ and again took the steps `E`, `E`, `E`, `S`, `S`, `W`. Once more it would finish in $(7, 3)$ where you are (this time it does not move during the second or third `E` because this would take it outside the forest). In fact the robot might have started in any of the locations $(7, 1)$, $(8, 1)$, $(5, 2)$, $(6, 2)$, $(7, 2)$, $(8, 2)$, $(7, 3)$ or $(8, 3)$, and it still would have finished where you are standing.

Note that the robot is allowed to start in the square where you are waiting, and/or may travel through your square any number of times along the way (you were probably elsewhere chatting with a monkey at the time). However, the robot may not start in a square containing a tree, and it may not start outside the forest.

Given the map of the forest, your current location, and the steps taken by the robot to get to you, your task is to find the number of possible locations at which the robot may have started its trek to find you.

Constraints

- $1 \leq w, h \leq 100$, where w is the width and h is the height of the map (measured in grid squares);
- $1 \leq p \leq 1\,000$, where p is the number of steps recorded by the robot.

Input

Your program must read from standard input. The first line will contain the integers w h separated by a single space, as described above. Following this will be h lines, each describing a row of the forest map. Each of these lines will contain w characters. These characters will be one of:

- '`.`' — an empty patch of ground;
- '`T`' — a tree;
- '`U`' — the location at which the robot finds you.

The character '`U`' will appear exactly once on the map.

This will be followed by a line containing the integer p, then p lines describing the steps the robot took to find you. Each of these lines will contain a single letter — `N`, `E`, `S` or `W` — corresponding to steps in the north, east, south or west directions, respectively. You are guaranteed that there is at least one possible starting point that will get the robot to you using this sequence of steps.

All letters in the input will be given in upper case.

Output

Your program must write a single line to standard output. This line must contain a single integer giving the number of possible locations at which the robot might have started.

Sample Input

```
8 6
.T...T..
T.......
..T..TU.
....T..T
...T....
..T.T.T.
6
E
E
E
S
S
W
```

Sample Output

```
8
```

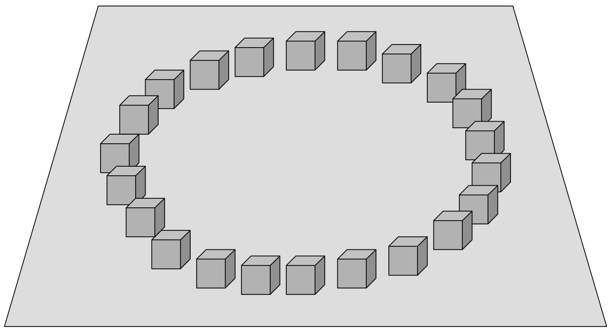
Scoring

The score for each input scenario will be 100% if the correct answer is output, or 0% otherwise.

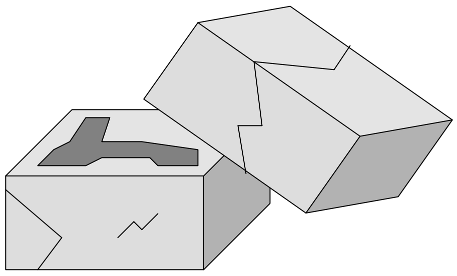
Problem 2
Secret Chamber of the Giza Pyramid

Time and Memory Limits: 3 seconds, 32 Mb

You are part of a team of archaeologists who have just discovered a secret chamber inside the great pyramid of Giza. Unfortunately, this chamber is completely empty, except for 22 small cubic stones forming a circle on the ground in the middle of the chamber.



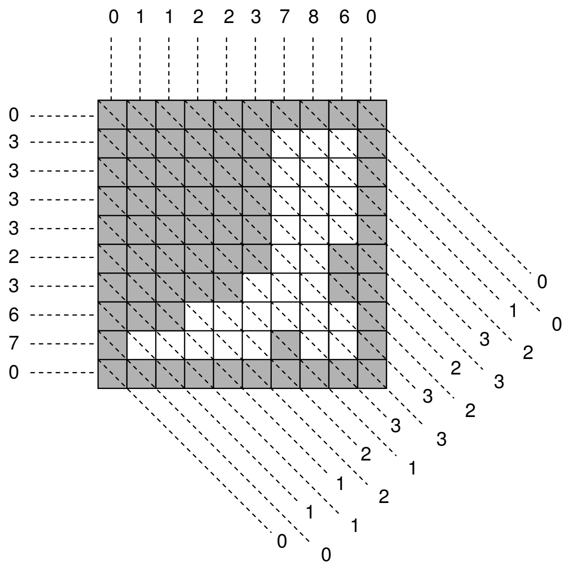
You notice that two of these cubes are broken, and after detailed observation, you realise that each of them has a hieroglyph carved in its central horizontal layer. A hidden message of great historical significance is waiting to be discovered. However, you can't break or even move the remaining 20 cubes to read the message, so you decide to use radiography, and take three X-ray pictures of each cube.



The results clearly show that, as you expected, a hieroglyph is carved in the middle horizontal layer of each cube, but since your X-ray pictures are taken from the sides of the cube and not from the top, you are unable to read the hidden hieroglyph directly. You need your computer to help you with this task.

The middle layer of each cube can be represented as an $N \times N$ grid of squares, which are either filled or empty. From your three pictures, you know the exact number of squares that are empty in each column, row and diagonal (the diagonals run in the top-left to bottom-right direction).

The grid squares are given (x, y) coordinates, so that the top-left square has coordinates $(0, 0)$ and the bottom-right square has coordinates $(N-1, N-1)$. The rows, columns and diagonals are each numbered $0, 1, 2, \ldots$. A square that has coordinates (x, y) is part of column x, line y, and diagonal $x - y + (N - 1)$; thus the short diagonal in the bottom-left corner is diagonal 0, the long diagonal running from square $(0, 0)$ to $(N-1, N-1)$ is diagonal $N - 1$, and the short diagonal in the top-right corner is diagonal $2N - 2$.



The diagram above shows an example hieroglyph with the results from the three X-rays, showing the number of empty squares in each column, row and diagonal.

Write a program that, given the number of empty squares in each column, row and diagonal of the grid, outputs a hieroglyph that matches the number of empty squares in each direction as closely as possible with the input.

Constraints

- $1 \leq N \leq 100$, where N is the side of the grid.

Input

Your program must read from standard input. The first line will contain the integer N, indicating the size of the $N \times N$ grid.

The second line will contain N space-separated integers. The ith of these integers gives the number of empty squares in the ith column of the grid, starting from the left.

The third line will contain N space-separated integers. The ith of these integers gives the number of empty squares in the ith row of the grid, starting from the top.

The fourth line will contain $(2N - 1)$ space-separated integers. The ith of these integers gives the number of empty squares in the ith diagonal of the grid, starting from the bottom left.

Output

Your program must write N lines to standard output, describing a possible hieroglyph. Each line must contain N characters describing a row of the grid, where the character '1' denotes a filled square, and '0' denotes an empty square. The output must not contain any spaces.

Sample Input

```
10
0 1 1 2 2 3 7 8 6 0
0 3 3 3 3 2 3 6 7 0
0 0 1 1 1 2 2 1 3 3 3 2 2 3 3 2 1 0 0
```

Sample Output

```
1111111111
1111110001
1111110001
1111110001
1111110001
1111110011
1111100011
1110000001
1000001001
1111111111
```

Scoring

For a given input scenario, the sums of empty squares on each column, row and diagonal of your output grid will be compared to the input values. The error for each column, row and diagonal will be the absolute difference between the input value and the value obtained from your grid. The total error of your output will be the sum of these errors. Your score for this input scenario will then be :

- 0%, if the total error of your output is larger than the total error obtained from a grid containing no empty squares;
- 100%, if the total error of your output is zero;
- otherwise, determined by a linear scale according to the total error, with the 0% and 100% marks on the scale corresponding to the cases described above.

Problem 3
Architecture

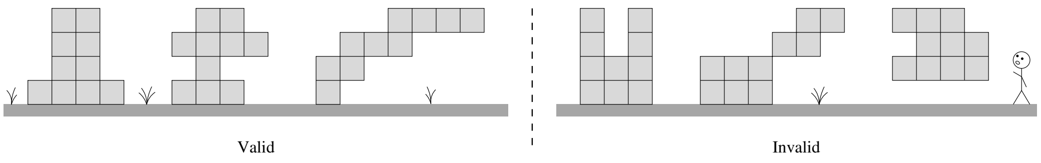
Time and Memory Limits: 2 seconds, 32 Mb

You are an avant-garde architect renowned for your beautiful but highly unstable designs. Your current project is to build an apartment complex with striking profiles and gorgeous views.

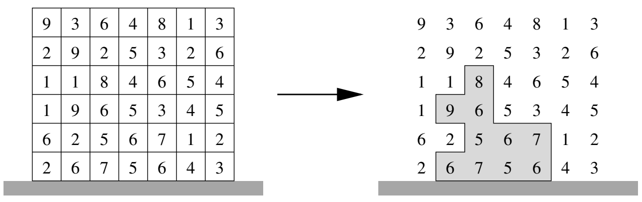
Your apartment complex must consist of a number of floors, where each floor contains a number of apartments. It is extremely important that:

- Each floor consists of a single connected row of apartments. Having separate disconnected sections of a floor is far too 1990s.
- At least one apartment from each floor must rest upon some apartment from the floor below it. The bottom floor must rest on the ground.

Some examples of valid and invalid buildings can be seen in the following diagram. Each of the buildings on the left hand side follows the rules above, and so is valid. All three buildings on the right hand side are invalid — in the first of these buildings the third floor contains two disconnected sections, in the second building the third floor does not rest upon the second, and in the final building the bottom floor does not touch the ground.



Unfortunately your creative desires are stifled by commercial constraints — your apartments will not sell unless they have the best possible views. For each point in space you have a number indicating how pretty the view is from that point. Your task is, given some number of apartments N, to design a building with precisely N apartments where the sum of these "prettiness numbers" for the N apartments is as large as possible.



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

For example, consider the grid on the left hand side of the diagram above. This gives the prettiness numbers for a strip of land, where the bottom row of numbers describe apartments on the bottom floor, the second row describes apartments on the second floor, and so on.

Suppose you are asked to create a building containing $N = 10$ apartments. An example of such a building is illustrated on the right hand side of the diagram. The sum of the prettiness numbers for this building is $8 + 9 + 6 + 5 + 6 + 7 + 6 + 7 + 5 + 6 = 65$. With a little investigation it can be seen that this is the best possible building that you can create, i.e., no sum larger than 65 is possible.

Constraints

- $1 \le N \le 80$, where N is the number of apartments in your building;
- $1 \le W, H \le 80$, where the grid of prettiness numbers has width W and height H;
- It is possible to construct a building with N apartments, i.e., $N \le W \times H$.

Furthermore, for 30% of the available marks, the input will satisfy $1 \le N, W, H \le 20$.

Input

Your program must read from standard input. The first line of input will contain the single integer N, the number of apartments that your building must contain.

The second line of input will be of the form $W\ H$, giving the dimensions of the grid of numbers. Following this will be H lines each with W space-separated integers, giving the prettiness numbers for the different points in space. The last row of input describes prettiness numbers for the ground floor, the second-last row describes the second floor, and so on. All prettiness numbers will be between 1 and 100 000 inclusive.

Output

Your program must write a single line to standard output. This line must contain a single integer, giving the largest possible sum of prettiness numbers for your building.

Sample Input

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

Sample Output

```
65
```

Scoring

The score for each input scenario will be 100% if the correct answer is output, or 0% otherwise.

Problem 4
Global Warming

Time and Memory Limits: 1 seconds, 16 Mb

Global warming is a rather *hot* topic these days, leading several scientists to try to model the behaviour of the climate for the coming decades. One of these scientists has devised a method which he claims can tell you, for a given point on earth, a number of predictions about the future climate at this location. Each prediction is of the form: "between day X and day Y, the temperature at this place will reach a maximum value of K degrees", where X, Y and K are parameters with integer values.

The scientist seems convinced that these predictions are absolutely accurate, while you are a little doubtful about this point. You wish to write a program to find out, given a place on earth, whether the set of predictions made by the scientist is coherent. By *coherent*, we mean it is possible to find an actual sequence of temperatures $M_1, M_2, \ldots M_N$ for which M_i is the maximal temperature on day i, and where all of the predictions are satisfied. A prediction (X, Y, K) is *satisfied* if, between day X and day Y (inclusive), the temperature never rises above K and there exists at least one day in that period where the temperature is exactly K.

Constraints

- $1 \leq T \leq 5$, where T is the number of places on earth at which you plan to test the scientist's method;
- $1 \leq N \leq 100\,000$, where N is the number of days being studied for any one place;
- $1 \leq P \leq 50\,000$, where P is the number of predictions made for any one place;
- $1 \leq X \leq Y \leq N$, where X and Y are days involved in a prediction (where 1 is the first day of the study period and N is the last);
- $1 \leq K \leq 100\,000$, where K is a predicted maximum temperature (measured in hundredths of degrees Kelvin).

Furthermore, for 30% of the available marks, each study period will satisfy $P \leq 100$.

Input

Your program must read from standard input. The first line will contain the single integer T, the number of places at which you wish to test the scientist's method. Following this will be a description of each of the T tests, one after another.

Each description will begin with a single line containing the two integers N P separated by a single space. Following this will be P lines each containing a single prediction. Each prediction will be described by three integers X Y K separated by spaces.

Output

Your program must write precisely T lines to standard output. The ith of these lines must contain a single integer, corresponding to the ith test description. For each test, this integer should be '1' if the corresponding predictions are coherent, or '0' if it is not possible to satisfy all of the predictions at this place.

Sample Input

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

Sample Output

```
1
0
```

Explanation

The sample data contains two test descriptions. The first test seeks to verify three predictions over a five day period. These predictions state that:

- out of days 1 and 2, one of these days must have a maximum temperature of 2, and none of these days should exceed 2;
- out of days 4 and 5, one of these days must have a maximum temperature of 1, and none of these days should exceed 1;
- out of days 2, 3 and 4, one of these days must have a maximum temperature of 3, and none of these days should exceed 3.

One such possibility that satisfies these predictions is the sequence of temperatures 1, 2, 3, 1, 1. Thus this set of predictions can be satisfied.

The second test seeks to verify four predictions over a four day period. These predictions are:

- day 3 must have a maximum temperature of 4;
- day 4 must have a maximum temperature of 3;
- out of days 1, 2, 3 and 4, one of these days must have a maximum temperature of 2, and none of these days should exceed 2;
- out of days 1 and 2, one of these days must have a maximum temperature of 1, and none of these days should exceed 1.

Clearly, the third statement conflicts with the first and second, and so this set of predictions cannot be satisfied.

Scoring

The score for each input scenario will be 100% if the correct sequence of answers is output, or 0% otherwise.