# Party

SEASON 5 - FINAL

After all difficulties during the last season, Ivancho has finally returned to ordinary life. He lives in a mountainous region, where his friends live as well. Some of them are preparing to organize P parties in their homes in the near future. Ivancho exits his home and is ready to visit some of the parties. For each party it is known how much time after Ivancho exits his home it begins and for how long it lasts.

Ivanch loves being the center of attention and that makes him happier at parties. That's why he has the habit of giving cake to the other guests. His happiness from a given party is $time * (cakes + 1)$, where $time$ is the time he has spent at the party and $cakes$ – the number of cakes he has treated the other guests with. Ivancho can buy from *K* bakeries in the region. He can buy an unlimited number of cakes overall and they don't cost anything to him. However, at every purchase, he can only buy 100 000 cakes at most. Buying cakes takes no time. When Ivancho arrives at a party, he can use any number of cakes, as long as he has at least that many at the moment. He can treat the others with all of his cakes, he can also not give any cakes and instead keep them for later.

When Ivancho reaches a place in which there is a party or there will be one, he can join it immediately if it has already begun or he can wait for its beginning, but once he has entered a party, he cannot leave it until it ends. After that he can wait for another party at the same place or go to another place. Help him gain maximum happiness from the parties by making a program **`party`**, which creates a schedule for him.

The place in which Ivancho lives can be represented as a square matrix (the rows and columns of which are indexed from one to N), where each cell has a given height. For each party the coordinates of the cell in which it is held are known and Ivancho's home is in the cell at row *A* and column *B*. He can move from cell to cell upwards, downwards, left and right and he is not allowed to leave the matrix. Movement between two cells takes time equal to : $(\Delta + cakes)^2 + 1$, where $\Delta$ is the absolute value of the difference between their heights and $cakes$ – the number of cakes Ivancho has at the moment.

It is guaranteed that in Ivancho's home there is never any party, there are never any parties in in a bakery, that there are no two parties active at the same moment in the same cell and that there is no bakery in Ivancho's home.

### Input

On the first line of the file `party.in` there will be the integers *N*, *P* and *K*. They are followed by *N* lines, each of which contains *N* integers – the heights of the terrain in each cell. On the next row there will be the coordinates *A* and *B* of the cell in which Ivancho's home is, and on the next *P* lines each of the parties is described using 4 integers – the coordinates (row and column) of the cell in which the party is held, its beginning and its duration. Each of the last *K* lines contains 2 integers each – the coordinates of a cell (row, followed by column), in which there is a bakery.

### Output

On the only line of the output file `party.out` the program must print a string. The string must represent a schedule, described by the uppercase Latin letters 'R', 'L', 'U', 'D', digits and the sign '+'. The letters mean that Ivancho has to move respectively right, left, up or down. A number, printed at a moment at which he is in a cell with a bakery shows the number of cakes he has to buy and the lack of a number – that he won't buy cakes. While he is in a cell, in which a party is going to begin or has already begun, printing the character '+' (plus) means that Ivancho has to

visit the party. If after the '+' sign there is a number, it shows how many cakes he will use. If a number if missing, Ivancho won't give any cakes away. It is possible that Ivancho visits many consecutive parties in the same cell. In this case the program must print a plus for each party, possibly followed by the number of cakes which Ivancho will give away.

Important! Your output must end before more than 10 000 000 000 ($10^{10}$) units of time have passed.

### Constraints

```
3 <= N <= 200
1 <= P <= 100 000
1 <= K <= N
0 <= height of each cell <= 99
0 <= duration of each party <= 10 000
End of the last party <= 1 000 000 000 units of time
```

### Grading

If the output for a given test is valid, you will receive $(\frac{yours+1}{max+1})^2 * 100\ \%$ of the points for it. The output is considered invalid under one of the following conditions:

- The output string has characters which are not mentioned in the task statement
- Ivancho leaves the matrix
- There is a '+' while Ivancho is at home, in a cell with no present or future party or in a cell with a bakery.
- Ivancho tries to buy cakes from a cell with no bakery
- Ivancho tries to give away more cakes than he has
- Ivancho tries to buy more than 100 000 cakes at once
- There is no output string

### Testing

Your solution will be tested on 12 preliminary (during the contest) and 40 final tests. They will be divided in 4 groups, each of which consists of 25% of all tests.

| Group 1 | Group 2 | Group 3 | Group 4 |
|---|---|---|---|
| 3<= N <= 20<br>1<= P <= 1000<br>1<= K <= N | 3<= N <= 100<br>1<= P <= 100 000<br>1<= K <= N | 3<= N <= 200<br>1<= P <= 10 000<br>1<= K <= N | 3<= N <= 200<br>1<= P <= 100 000<br>1<= K <= N |

Time limit (TL) – 5 sec

Memory limit (ML) – 256 MB

### Test generator

At the site of the contest a test generator will be published. With it you can generate tests with which to test your own program. It receives either the number of a test group, from which the generated test will be, or custom *N*, *P* and *K*.

### Checking the validity of your output

There is also a checker available, with which the contestants can see how many points a given solution gets. It works in 4 modes:

1 – evaluates files with names `party.in` and `party.out`, which have to be in the same folder as the checker.

2 – evaluates files with names given by you, which have to be in the same folder as the checker.

3 – evaluates files with names `party.**.in` and `party.**.out`, where you specify in the standard input the first and the last number, which are in the place of the asterisks, and it prints all results in a file with the name `party.log`.

4 – executes your program at T tests, generated by the test generator, where the program being tested reads from the file `party.in` and writes in the file `party.out`. The solution being executed has to be called `party.exe`. The checker will write the results in a file with the name `party.log`. All generated tests will be saved in the folder with name `party-tests`.

### Example tests

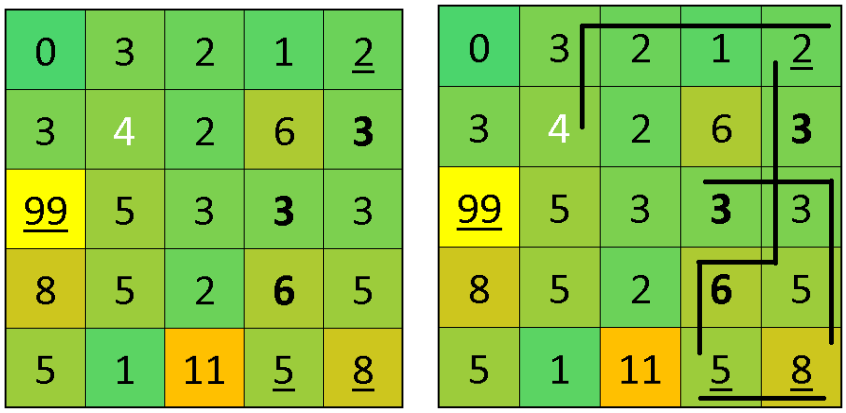
| Input (party.in) | Output (party.out) |
|---|---|
| 10 8 3<br>1 2 3 2 1 2 3 4 5 6<br>9 8 7 6 5 4 3 2 1 0<br>5 5 5 5 5 5 5 5 5 5<br>4 3 2 1 0 1 2 3 4 5<br>9 9 9 8 8 8 2 7 7 6<br>9 9 9 6 6 5 5 4 4 3<br>9 8 7 3 2 2 2 1 1 1<br>6 5 4 0 0 0 1 2 3 4<br>0 0 0 5 6 7 8 9 8 7<br>0 0 0 6 5 4 3 2 1 0<br>6 6<br>5 5 3 2<br>7 1 2 2<br>10 10 0 12<br>10 10 500 150<br>10 10 1000 250<br>10 10 2000 150<br>10 10 3000 150<br>2 3 124 444<br>9 9<br>6 10<br>6 7 | RRRDDD5RD+5UL8DR+8ULDR++ |
| 5 4 3<br>0 3 2 1 2<br>3 4 2 6 3<br>99 5 3 3 3<br>8 5 2 6 5<br>5 1 11 5 8<br>2 2<br>3 1 999 999<br>5 5 120 20<br>5 4 42 69<br>1 5 7 2<br>3 4 | URRR+DDDL3D+2R+1UUL99999 |

| | |
|---|---|
| 2 5<br>4 4 | |

### Explanation of the first example

The example output brings 3360 happiness to Ivancho.

### Explanation of the second example




In the second image the path Ivancho has to follow is shows with a solid black line. His home is indicated with white text. The interruptions of the path line mean that Ivancho has visited a party in the given cell. In the zones with underlined height there is a party and in the ones with a bold font – bakeries.