Educational Codeforces Round 80 (Rated for Div. 2)

A. Deadline

2 seconds, 256 megabytes

Adilbek was assigned to a special project. For Adilbek it means that he has n days to run a special program and provide its results. But there is a problem: the program needs to run for d days to calculate the results.

Fortunately, Adilbek can optimize the program. If he spends x (x is a non-negative integer) days optimizing the program, he will make the program run in $\left\lceil \frac{d}{x+1} \right\rceil$ days ($\lceil a \rceil$ is the ceiling function: $\lceil 2.4 \rceil = 3$, $\lceil 2 \rceil = 2$). The program cannot be run and optimized simultaneously, so the total number of days he will spend is equal to $x + \left\lceil \frac{d}{x+1} \right\rceil$.

Will Adilbek be able to provide the generated results in no more than n days?

Input

The first line contains a single integer T ($1 \le T \le 50$) — the number of test cases.

The next T lines contain test cases – one per line. Each line contains two integers n and d ($1 \le n \le 10^9$, $1 \le d \le 10^9$) — the number of days before the deadline and the number of days the program runs.

Output

Print T answers — one per test case. For each test case print `YES` (case insensitive) if Adilbek can fit in n days or `NO` (case insensitive) otherwise.

input

```
3
1 1
4 5
5 11
```

output

```
YES
YES
NO
```

In the first test case, Adilbek decides not to optimize the program at all, since $d \le n$.

In the second test case, Adilbek can spend 1 day optimizing the program and it will run $\left\lceil \frac{5}{2} \right\rceil = 3$ days. In total, he will spend 4 days and will fit in the limit.

In the third test case, it's impossible to fit in the limit. For example, if Adilbek will optimize the program 2 days, it'll still work $\left\lceil \frac{11}{2+1} \right\rceil = 4$ days.

B. Yet Another Meme Problem

1 second, 256 megabytes

Try guessing the statement from this picture http://tiny.cc/ogyoiz.

You are given two integers A and B, calculate the number of pairs (a, b) such that $1 \le a \le A$, $1 \le b \le B$, and the equation $a \cdot b + a + b = conc(a, b)$ is true; $conc(a, b)$ is the concatenation of a and b (for example, $conc(12, 23) = 1223$, $conc(100, 11) = 10011$). a **and** b **should not contain leading zeroes**.

Input

The first line contains t ($1 \le t \le 100$) — the number of test cases.

Each test case contains two integers A and B $(1 \le A, B \le 10^9)$.

Output

Print one integer — the number of pairs (a, b) such that $1 \le a \le A$, $1 \le b \le B$, and the equation $a \cdot b + a + b = conc(a, b)$ is true.

input

```
3
1 11
4 2
191 31415926
```

output

```
1
0
1337
```

There is only one suitable pair in the first test case: $a = 1, b = 9$ ($1 + 9 + 1 \cdot 9 = 19$).

C. Two Arrays

1 second, 256 megabytes

You are given two integers n and m. Calculate the number of pairs of arrays (a, b) such that:

- the length of both arrays is equal to m;
- each element of each array is an integer between 1 and n (inclusive);
- $a_i \le b_i$ for any index i from 1 to m;
- array a is sorted in non-descending order;
- array b is sorted in non-ascending order.

As the result can be very large, you should print it modulo $10^9 + 7$.

Input

The only line contains two integers n and m ($1 \le n \le 1000$, $1 \le m \le 10$).

Output

Print one integer – the number of arrays a and b satisfying the conditions described above modulo $10^9 + 7$.

input

```
2 2
```

output

```
5
```

input

```
10 1
```

output

```
55
```

input

```
723 9
```

output

```
157557417
```

In the first test there are 5 suitable arrays:

- $a = [1, 1], b = [2, 2]$;
- $a = [1, 2], b = [2, 2]$;
- $a = [2, 2], b = [2, 2]$;
- $a = [1, 1], b = [2, 1]$;
- $a = [1, 1], b = [1, 1]$.

D. Minimax Problem

5 seconds, 512 megabytes

You are given n arrays a_1, a_2, ..., a_n; each array consists of exactly m integers. We denote the y-th element of the x-th array as $a_{x,y}$.

You have to choose two arrays a_i and a_j ($1 \le i, j \le n$, it is possible that $i = j$). After that, you will obtain a new array b consisting of m integers, such that for every $k \in [1, m]$ $b_k = \max(a_{i,k}, a_{j,k})$.

Your goal is to choose i and j so that the value of $\min\limits_{k=1}^{m} b_k$ is maximum possible.

Input

The first line contains two integers n and m ($1 \le n \le 3 \cdot 10^5$, $1 \le m \le 8$) — the number of arrays and the number of elements in each array, respectively.

Then n lines follow, the x-th line contains the array a_x represented by m integers $a_{x,1}$, $a_{x,2}$, ..., $a_{x,m}$ ($0 \le a_{x,y} \le 10^9$).

Output

Print two integers i and j ($1 \le i, j \le n$, **it is possible that $i = j$**) — the indices of the two arrays you have to choose so that the value of $\min\limits_{k=1}^{m} b_k$ is maximum possible. If there are multiple answers, print any of them.

input
```
6 5
5 0 3 1 2
1 8 9 1 3
1 2 3 4 5
9 1 0 3 7
2 3 0 6 3
6 4 1 7 0
```
output
```
1 5
```

E. Messenger Simulator

3 seconds, 256 megabytes

Polycarp is a frequent user of the very popular messenger. He's chatting with his friends all the time. He has n friends, numbered from 1 to n.

Recall that a permutation of size n is an array of size n such that each integer from 1 to n occurs exactly once in this array.

So his recent chat list can be represented with a permutation p of size n. p_1 is the most recent friend Polycarp talked to, p_2 is the second most recent and so on.

Initially, Polycarp's recent chat list p looks like $1, 2, \ldots, n$ (in other words, it is an identity permutation).

After that he receives m messages, the j-th message comes from the friend a_j. And that causes friend a_j to move to the first position in a permutation, shifting everyone between the first position and the current position of a_j by 1. Note that if the friend a_j is in the first position already then nothing happens.

For example, let the recent chat list be $p = [4, 1, 5, 3, 2]$:

- if he gets messaged by friend 3, then p becomes $[3, 4, 1, 5, 2]$;
- if he gets messaged by friend 4, then p doesn't change $[4, 1, 5, 3, 2]$;
- if he gets messaged by friend 2, then p becomes $[2, 4, 1, 5, 3]$.

For each friend consider all position he has been at in the beginning and after receiving each message. Polycarp wants to know what were the minimum and the maximum positions.

Input

The first line contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) — the number of Polycarp's friends and the number of received messages, respectively.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$) — the descriptions of the received messages.

Output

Print n pairs of integers. For each friend output the minimum and the maximum positions he has been in the beginning and after receiving each message.

input
```
5 4
3 5 1 4
```
output
```
1 3
2 5
1 4
1 5
1 5
```

input
```
4 3
1 2 4
```
output
```
1 3
1 2
3 4
1 4
```

In the first example, Polycarp's recent chat list looks like this:

- $[1, 2, 3, 4, 5]$
- $[3, 1, 2, 4, 5]$
- $[5, 3, 1, 2, 4]$
- $[1, 5, 3, 2, 4]$
- $[4, 1, 5, 3, 2]$

So, for example, the positions of the friend 2 are $2, 3, 4, 4, 5$, respectively. Out of these 2 is the minimum one and 5 is the maximum one. Thus, the answer for the friend 2 is a pair $(2, 5)$.

In the second example, Polycarp's recent chat list looks like this:

- $[1, 2, 3, 4]$
- $[1, 2, 3, 4]$
- $[2, 1, 3, 4]$
- $[4, 2, 1, 3]$

F. Red-Blue Graph

1 second, 512 megabytes

You are given a bipartite graph: the first part of this graph contains n_1 vertices, the second part contains n_2 vertices, and there are m edges. **The graph can contain multiple edges**.

Initially, each edge is colorless. For each edge, you may either leave it uncolored (it is free), paint it red (it costs r coins) or paint it blue (it costs b coins). No edge can be painted red and blue simultaneously.

There are three types of vertices in this graph — colorless, red and blue. Colored vertices impose additional constraints on edges' colours:

- for each red vertex, the number of red edges indicent to it should be **strictly greater** than the number of blue edges incident to it;
- for each blue vertex, the number of blue edges indicent to it should be **strictly greater** than the number of red edges incident to it.

Colorless vertices impose no additional constraints.

Your goal is to paint some (possibly none) edges so that all constraints are met, and among all ways to do so, you should choose the one with minimum total cost.

Input

The first line contains five integers n_1, n_2, m, r and b ($1 \le n_1, n_2, m, r, b \le 200$) — the number of vertices in the first part, the number of vertices in the second part, the number of edges, the amount of coins you have to pay to paint an edge red, and the amount of coins you have to pay to paint an edge blue, respectively.

The second line contains one string consisting of n_1 characters. Each character is either U, R or B. If the i-th character is U, then the i-th vertex of the first part is uncolored; R corresponds to a red vertex, and B corresponds to a blue vertex.

The third line contains one string consisting of n_2 characters. Each character is either U, R or B. This string represents the colors of vertices of the second part in the same way.

Then m lines follow, the i-th line contains two integers u_i and v_i ($1 \le u_i \le n_1$, $1 \le v_i \le n_2$) denoting an edge connecting the vertex u_i from the first part and the vertex v_i from the second part.

The graph may contain multiple edges.

Output

If there is no coloring that meets all the constraints, print one integer -1.

Otherwise, print an integer c denoting the total cost of coloring, and a string consisting of m characters. The i-th character should be U if the i-th edge should be left uncolored, R if the i-th edge should be painted red, or B if the i-th edge should be painted blue. If there are multiple colorings with minimum possible cost, print any of them.

input

```
3 2 6 10 15
RRB
UB
3 2
2 2
1 2
1 1
2 1
1 1
```

output

```
35
BUURRU
```

input

```
3 1 3 4 5
RRR
B
2 1
1 1
3 1
```

output

```
-1
```

input

```
3 1 3 4 5
URU
B
2 1
1 1
3 1
```

output

```
14
RBB
```