Codeforces Round #616 (Div. 1)

A. Mind Control

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You and your $n-1$ friends have found an array of integers $a_1, a_2, \ldots, a_n$. You have decided to share it in the following way: All n of you stand in a line in a particular order. Each minute, the person at the front of the line chooses either the first or the last element of the array, removes it, and keeps it for himself. He then gets out of line, and the next person in line continues the process.

You are standing in the m-th position in the line. **Before the process starts**, you may choose up to k different people in the line, and persuade them to always take either the first or the last element in the array on their turn (for each person his own choice, not necessarily equal for all people), no matter what the elements themselves are. **Once the process starts, you cannot persuade any more people, and you cannot change the choices for the people you already persuaded**.

Suppose that you're doing your choices optimally. What is the greatest integer x such that, no matter what are the choices of the friends you didn't choose to control, the element you will take from the array will be greater than or equal to x?

Please note that the friends you don't control may do their choice **arbitrarily**, and they will not necessarily take the biggest element available.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three space-separated integers n, m and k ($1 \leq m \leq n \leq 3500$, $0 \leq k \leq n-1$) — the number of elements in the array, your position in line and the number of people whose choices you can fix.

The second line of each test case contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed 3500.

Output

For each test case, print the largest integer x such that you can guarantee to obtain at least x.

Example

input
4 6 4 2 2 9 2 3 8 5 4 4 1 2 13 60 4 4 1 3 1 2 2 1 2 2 0 1 2

output
8 4 1 1

Note

In the first test case, an optimal strategy is to force the first person to take the last element and the second person to take the first element.

- the first person will take the last element (5) because he or she was forced by you to take the last element. After this turn the remaining array will be $[2, 9, 2, 3, 8]$;
- the second person will take the first element (2) because he or she was forced by you to take the first element. After this turn the remaining array will be $[9, 2, 3, 8]$;
- if the third person will choose to take the first element (9), at your turn the remaining array will be $[2, 3, 8]$ and you will take 8 (the last element);
- if the third person will choose to take the last element (8), at your turn the remaining array will be $[9, 2, 3]$ and you will take 9 (the first element).

Thus, this strategy guarantees to end up with at least 8. We can prove that there is no strategy that guarantees to end up with at least 9. Hence, the answer is 8.

In the second test case, an optimal strategy is to force the first person to take the first element. Then, in the worst case, both the second and the third person will take the first element: you will end up with 4.

B. Irreducible Anagrams

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call two strings s and t *anagrams* of each other if it is possible to rearrange symbols in the string s to get a string, equal to t.

Let's consider two strings s and t **which are anagrams of each other**. We say that t is a *reducible anagram* of s if there exists an integer $k \geq 2$ and $2k$ non-empty strings $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ that satisfy the following conditions:

1. If we write the strings $s_1, s_2, \ldots, s_k$ in order, the resulting string will be equal to s;
2. If we write the strings $t_1, t_2, \ldots, t_k$ in order, the resulting string will be equal to t;
3. For all integers i between 1 and k inclusive, s_i and t_i are anagrams of each other.

If such strings don't exist, then t is said to be an *irreducible anagram* of s. **Note that these notions are only defined when s and t are anagrams of each other**.

For example, consider the string $s =$ "gamegame". Then the string $t =$ "megamage" is a reducible anagram of s, we may choose for example $s_1 =$ "game", $s_2 =$ "gam", $s_3 =$ "e" and $t_1 =$ "mega", $t_2 =$ "mag", $t_3 =$ "e":

$$s = \overbrace{\texttt{game}}^{s_1} | \overbrace{\texttt{gam}}^{s_2} | \overbrace{\texttt{e}}^{s_3}$$

$$t = \underbrace{\texttt{mega}}_{t_1} | \underbrace{\texttt{mag}}_{t_2} | \underbrace{\texttt{e}}_{t_3}$$

On the other hand, we can prove that $t =$ "memegaga" is an irreducible anagram of s.

You will be given a string s and q queries, represented by two integers $1 \le l \le r \le |s|$ (where $|s|$ is equal to the length of the string s). For each query, you should find if the substring of s formed by characters from the l-th to the r-th has at least one irreducible anagram.

Input
The first line contains a string s, consisting of lowercase English characters ($1 \le |s| \le 2 \cdot 10^5$).

The second line contains a single integer q ($1 \le q \le 10^5$) — the number of queries.

Each of the following q lines contain two integers l and r ($1 \le l \le r \le |s|$), representing a query for the substring of s formed by characters from the l-th to the r-th.

Output
For each query, print a single line containing "Yes" (without quotes) if the corresponding substring has at least one irreducible anagram, and a single line containing "No" (without quotes) otherwise.

Examples

input
aaaaa 3 1 1 2 4 5 5
output
Yes No Yes

input
aabbbbbbc 6 1 2 2 4 2 2 1 9 5 7 3 5
output
No Yes Yes Yes No No

Note
In the first sample, in the first and third queries, the substring is "a", which has itself as an irreducible anagram since two or more non-empty strings cannot be put together to obtain "a". On the other hand, in the second query, the substring is "aaa", which has no irreducible anagrams: its only anagram is itself, and we may choose $s_1 =$ "a", $s_2 =$ "aa", $t_1 =$ "a", $t_2 =$ "aa" to show that it is a reducible anagram.

In the second query of the second sample, the substring is "abb", which has, for example, "bba" as an irreducible anagram.

C. Prefix Enlightenment

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n lamps on a line, numbered from 1 to n. Each one has an initial state off (0) or on (1).

You're given k subsets $A_1, \ldots, A_k$ of $\{1, 2, \ldots, n\}$, such that the intersection of any three subsets is empty. In other words, for all $1 \le i_1 < i_2 < i_3 \le k$, $A_{i_1} \cap A_{i_2} \cap A_{i_3} = \varnothing$.

In one operation, you can choose one of these k subsets and switch the state of all lamps in it. It is guaranteed that, with the given subsets, it's possible to make all lamps be simultaneously on using this type of operation.

Let m_i be the minimum number of operations you have to do in order to make the i first lamps be simultaneously on. Note that there is no condition upon the state of other lamps (between $i+1$ and n), they can be either off or on.

You have to compute m_i for all $1 \le i \le n$.

Input
The first line contains two integers n and k ($1 \le n, k \le 3 \cdot 10^5$).

The second line contains a binary string of length n, representing the initial state of each lamp (the lamp i is off if $s_i = 0$, on if $s_i = 1$).

The description of each one of the k subsets follows, in the following format:

The first line of the description contains a single integer c ($1 \le c \le n$) — the number of elements in the subset.

The second line of the description contains c distinct integers $x_1, \ldots, x_c$ ($1 \le x_i \le n$) — the elements of the subset.

It is guaranteed that:

- The intersection of any three subsets is empty;
- It's possible to make all lamps be simultaneously on using some operations.

Output
You must output n lines. The i-th line should contain a single integer m_i — the minimum number of operations required to make the lamps 1 to i be simultaneously on.

Examples

input

```
7 3
0011100
3
1 4 6
3
3 4 7
2
2 3
```

output

```
1
2
3
3
3
3
3
```

input

```
8 6
00110011
3
1 3 8
5
1 2 5 6 7
2
6 8
2
3 5
2
4 7
1
2
```

output

```
1
1
1
1
1
1
4
4
```

input

```
5 3
00011
3
1 2 3
1
4
3
3 4 5
```

output

```
1
1
1
1
1
```

input

```
19 5
1001001001100000110
2
2 3
2
5 6
2
8 9
5
12 13 14 15 16
1
19
```

output

```
0
1
1
1
2
2
2
3
3
3
3
4
4
4
4
4
4
4
5
```

Note

In the first example:

- For $i = 1$, we can just apply one operation on A_1, the final states will be 1010110;
- For $i = 2$, we can apply operations on A_1 and A_3, the final states will be 1100110;
- For $i \geq 3$, we can apply operations on A_1, A_2 and A_3, the final states will be 1111111.

In the second example:

- For $i \leq 6$, we can just apply one operation on A_2, the final states will be 11111101;
- For $i \geq 7$, we can apply operations on A_1, A_3, A_4, A_6, the final states will be 11111111.

D. Coffee Varieties (hard version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the hard version of the problem. You can find the easy version in the Div. 2 contest. Both versions only differ in the number of times you can ask your friend to taste coffee.

This is an interactive problem.

You're considering moving to another city, where one of your friends already lives. There are n cafés in this city, where n is a power of two. The i-th café produces a single variety of coffee a_i.

As you're a coffee-lover, before deciding to move or not, **you want to know the number d of distinct varieties of coffees** produced in this city.

You don't know the values $a_1, \ldots, a_n$. Fortunately, your friend has a memory of size k, where k is a power of two.

Once per day, you can ask him to taste a cup of coffee produced by the café c, and he will tell you if he tasted a similar coffee during the last k days.

You can also ask him to take a medication that will reset his memory. He will forget all previous cups of coffee tasted. You can reset his memory at most $30\ 000$ times.

More formally, the memory of your friend is a queue S. Doing a query on café c will:

- Tell you if a_c is in S;
- Add a_c at the back of S;
- If $|S| > k$, pop the front element of S.

Doing a reset request will pop all elements out of S.

Your friend can taste at most $\frac{3n^2}{2k}$ cups of coffee in total. Find the diversity d (number of distinct values in the array a).

Note that asking your friend to reset his memory **does not count** towards the number of times you ask your friend to taste a cup of coffee.

In some test cases the behavior of the interactor **is adaptive**. It means that the array a may be **not fixed** before the start of the interaction and may **depend on your queries**. It is guaranteed that at any moment of the interaction, there is at least one array a consistent with all the answers given so far.

Input

The first line contains two integers n and k ($1 \le k \le n \le 1024$, k and n are powers of two).

It is guaranteed that $\frac{3n^2}{2k} \le 15\ 000$.

Interaction

You begin the interaction by reading n and k.

- To ask your friend to taste a cup of coffee produced by the café c, in a separate line output
  `? c`

  Where c must satisfy $1 \le c \le n$. Don't forget to flush, to get the answer.

  In response, you will receive a single letter `Y` (yes) or `N` (no), telling you if variety a_c is one of the last k varieties of coffee in his memory.

- To reset the memory of your friend, in a separate line output the single letter `R` **in upper case**. You can do this operation at most $30\ 000$ times.
- When you determine the number d of different coffee varieties, output
  `! d`

In case your query is invalid, you asked more than $\frac{3n^2}{2k}$ queries of type `?` or you asked more than $30\ 000$ queries of type `R`, the program will print the letter `E` and will finish interaction. You will receive a **Wrong Answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get **Idleness limit exceeded**. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hack format

The first line should contain the word `fixed`

The second line should contain two integers n and k, separated by space ($1 \le k \le n \le 1024$, k and n are powers of two).

It must hold that $\frac{3n^2}{2k} \le 15\ 000$.

The third line should contain n integers $a_1, a_2, \ldots, a_n$, separated by spaces ($1 \le a_i \le n$).

Examples

input
4 2 N N Y N N N N

output
? 1 ? 2 ? 3 ? 4 R ? 4 ? 1 ? 2 ! 3

input
8 8 N N N N Y Y

output
? 2 ? 6 ? 4 ? 5 ? 2 ? 5 ! 6

Note

In the first example, the array is $a = [1, 4, 1, 3]$. The city produces 3 different varieties of coffee (1, 3 and 4).

The successive varieties of coffee tasted by your friend are $1, 4, \mathbf{1}, 3, 3, 1, 4$ (bold answers correspond to Y answers). Note that between the two ? 4 asks, there is a reset memory request R, so the answer to the second ? 4 ask is N. Had there been no reset memory request, the answer to the second ? 4 ask is Y.

In the second example, the array is $a = [1, 2, 3, 4, 5, 6, 6, 6]$. The city produces 6 different varieties of coffee.

The successive varieties of coffee tasted by your friend are $2, 6, 4, 5, \mathbf{2}, \mathbf{5}$.

E. Cartesian Tree

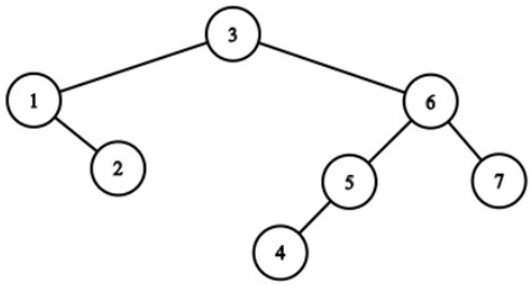
time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ildar is the algorithm teacher of William and Harris. Today, Ildar is teaching Cartesian Tree. However, Harris is sick, so Ildar is only teaching William.

A cartesian tree is a rooted tree, that can be constructed from a sequence of distinct integers. We build the cartesian tree as follows:

1. If the sequence is empty, return an empty tree;
2. Let the position of the **maximum** element be x;
3. Remove element on the position x from the sequence and break it into the left part and the right part (which might be empty) (not actually removing it, just taking it away temporarily);
4. Build cartesian tree for each part;
5. Create a new vertex for the element, that was on the position x which will serve as the root of the new tree. Then, for the root of the left part and right part, if exists, will become the children for this vertex;
6. Return the tree we have gotten.

For example, this is the cartesian tree for the sequence $4, 2, 7, 3, 5, 6, 1$:



After teaching what the cartesian tree is, Ildar has assigned homework. He starts with an empty sequence a.

In the i-th round, he inserts an element with value i somewhere in a. Then, he asks a question: what is the sum of the sizes of the subtrees for every node in the cartesian tree for the current sequence a?

Node v is in the node u subtree if and only if $v = u$ or v is in the subtree of one of the vertex u children. The size of the subtree of node u is the number of nodes v such that v is in the subtree of u.

Ildar will do n rounds in total. The homework is the sequence of answers to the n questions.

The next day, Ildar told Harris that he has to complete the homework as well. Harris obtained the final state of the sequence a from William. However, he has no idea how to find the answers to the n questions. Help Harris!

Input

The first line contains a single integer n ($1 \le n \le 150000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$). It is guarenteed that each integer from 1 to n appears in the sequence exactly once.

Output

Print n lines, i-th line should contain a single integer — the answer to the i-th question.

Examples

input
5 2 4 1 5 3

output
1 3 6 8 11

input
6 1 2 4 5 6 3

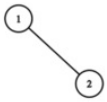
output
1 3 6 8 12 17

Note

After the first round, the sequence is 1. The tree is

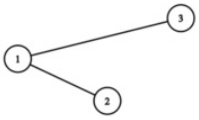
The answer is 1.

After the second round, the sequence is $2, 1$. The tree is



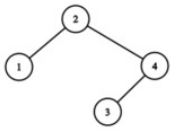
The answer is $2 + 1 = 3$.

After the third round, the sequence is $2, 1, 3$. The tree is



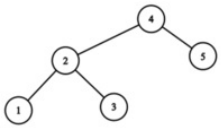
The answer is $2 + 1 + 3 = 6$.

After the fourth round, the sequence is $2, 4, 1, 3$. The tree is



The answer is $1 + 4 + 1 + 2 = 8$.

After the fifth round, the sequence is $2, 4, 1, 5, 3$. The tree is



The answer is $1 + 3 + 1 + 5 + 1 = 11$.

F. Making Shapes

time limit per test: 5 seconds
memory limit per test: 768 megabytes
input: standard input
output: standard output

You are given n pairwise non-collinear two-dimensional vectors. You can make shapes in the two-dimensional plane with these vectors in the following fashion:

1. Start at the origin $(0, 0)$.
2. Choose a vector and add the segment of the vector to the current point. For example, if your current point is at (x, y) and you choose the vector (u, v), draw a segment from your current point to the point at $(x + u, y + v)$ and set your current point to $(x + u, y + v)$.
3. Repeat step 2 until you reach the origin again.

You can reuse a vector as many times as you want.

Count the number of different, non-degenerate (with an area greater than 0) and convex shapes made from applying the steps, such that the shape can be contained within a $m \times m$ square, and the vectors building the shape are in counter-clockwise fashion. Since this number can be too large, you should calculate it by modulo 998244353.

Two shapes are considered the same if there exists some parallel translation of the first shape to another.

A shape can be contained within a $m \times m$ square if there exists some parallel translation of this shape so that every point (u, v) inside or on the border of the shape satisfies $0 \leq u, v \leq m$.

Input

The first line contains two integers n and m — the number of vectors and the size of the square ($1 \leq n \leq 5$, $1 \leq m \leq 10^9$).

Each of the next n lines contains two integers x_i and y_i — the x-coordinate and y-coordinate of the i-th vector ($|x_i|, |y_i| \leq 4$, $(x_i, y_i) \neq (0, 0)$).

It is guaranteed, that no two vectors are parallel, so for any two indices i and j such that $1 \leq i < j \leq n$, there is no real value k such that $x_i \cdot k = x_j$ and $y_i \cdot k = y_j$.

Output

Output a single integer — the number of satisfiable shapes by modulo 998244353.

Examples

input
3 3 -1 0 1 1 0 -1
output
3

input
3 3

-1 0
2 2
0 -1

output

1

input

3 1776966
-1 0
3 3
0 -2

output

296161

input

4 15
-4 -4
-1 1
-1 -4
4 3

output

1

input

5 10
3 -4
4 -3
1 -3
2 -3
-3 -4

output

0

input

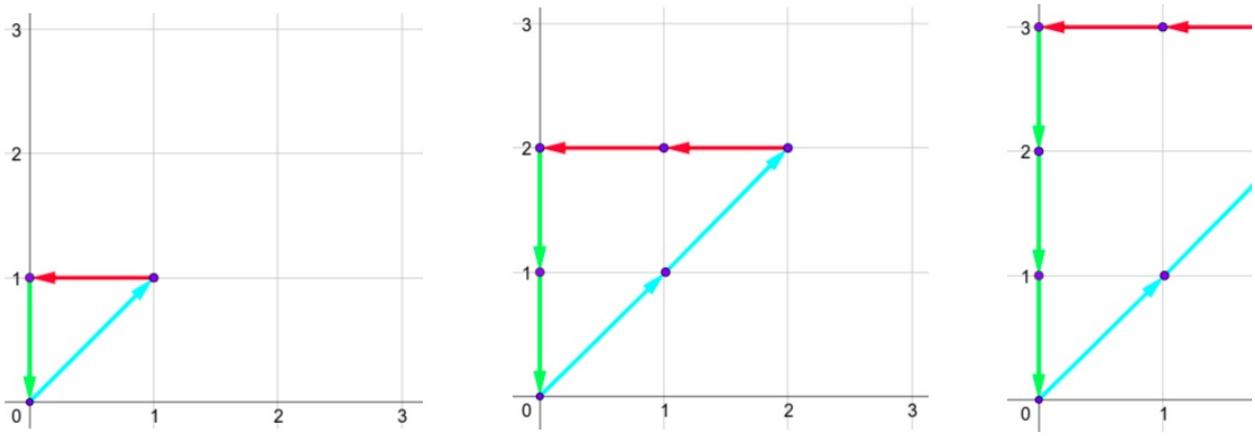
5 1000000000
-2 4
2 -3
0 -4
2 4
-1 -3

output

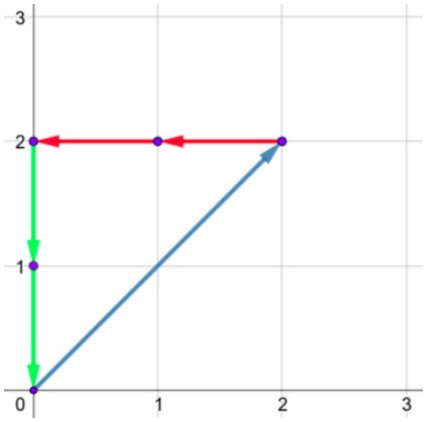
9248783

Note

The shapes for the first sample are:



The only shape for the second sample is:



The only shape for the fourth sample is: