# Educational Codeforces Round 123 (Rated for Div. 2)

## A. Doors and Keys

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

The knight is standing in front of a long and narrow hallway. A princess is waiting at the end of it.

In a hallway there are three doors: a red door, a green door and a blue door. The doors are placed one after another, however, possibly in a different order. To proceed to the next door, the knight must first open the door before.

Each door can be only opened with a key of the corresponding color. So three keys: a red key, a green key and a blue key — are also placed somewhere in the hallway. To open the door, the knight should first pick up the key of its color.

The knight has a map of the hallway. It can be transcribed as a string, consisting of six characters:

- R, G, B — denoting red, green and blue doors, respectively;
- r, g, b — denoting red, green and blue keys, respectively.

Each of these six characters appears in the string exactly once.

The knight is standing at the beginning of the hallway — on the left on the map.

Given a map of the hallway, determine if the knight can open all doors and meet the princess at the end of the hallway.

### Input

The first line contains a single integer $t$ $(1 \le t \le 720)$ — the number of testcases.

Each testcase consists of a single string. Each character is one of R, G, B (for the doors), r, g, b (for the keys), and each of them appears exactly once.

### Output

For each testcase, print YES if the knight can open all doors. Otherwise, print NO.

### Example

**input**

```
4
rgbBRG
RgbrBG
bBrRgG
rgRGBb
```

**output**

```
YES
NO
YES
NO
```

### Note

In the first testcase, the knight first collects all keys, then opens all doors with them.

In the second testcase, there is a red door right in front of the knight, but he doesn't have a key for it.

In the third testcase, the key to each door is in front of each respective door, so the knight collects the key and uses it immediately three times.

In the fourth testcase, the knight can't open the blue door.

## B. Anti-Fibonacci Permutation

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

Let's call a permutation $p$ of length $n$ **anti-Fibonacci** if the condition $p_{i-2} + p_{i-1} \ne p_i$ holds for all $i$ $(3 \le i \le n)$. Recall that the permutation is the array of length $n$ which contains each integer from $1$ to $n$ exactly once.

Your task is for a given number $n$ print $n$ **distinct** anti-Fibonacci permutations of length $n$.

### Input

The first line contains a single integer $t$ ($1 \le t \le 48$) — the number of test cases.

The single line of each test case contains a single integer $n$ ($3 \le n \le 50$).

### Output

For each test case, print $n$ lines. Each line should contain an anti-Fibonacci permutation of length $n$. In each test case, you cannot print any permutation more than once.

If there are multiple answers, print any of them. It can be shown that it is always possible to find $n$ different anti-Fibonacci permutations of size $n$ under the constraints of the problem.

**Example**

**input**

```
2
4
3
```

**output**

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

## C. Increase Subarray Sums

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

You are given an array $a_1, a_2, \dots, a_n$, consisting of $n$ integers. You are also given an integer value $x$.

Let $f(k)$ be the maximum sum of a contiguous subarray of $a$ after applying the following operation: add $x$ to the elements on exactly $k$ **distinct** positions. An empty subarray should also be considered, it has sum $0$.

Note that the subarray doesn't have to include all of the increased elements.

Calculate the maximum value of $f(k)$ for all $k$ from $0$ to $n$ independently.

### Input

The first line contains a single integer $t$ ($1 \le t \le 5000$) — the number of testcases.

The first line of the testcase contains two integers $n$ and $x$ ($1 \le n \le 5000$; $0 \le x \le 10^5$) — the number of elements in the array and the value to add.

The second line contains $n$ integers $a_1, a_2, \dots, a_n$ ($-10^5 \le a_i \le 10^5$).

The sum of $n$ over all testcases doesn't exceed $5000$.

### Output

For each testcase, print $n + 1$ integers — the maximum value of $f(k)$ for all $k$ from $0$ to $n$ independently.

**Example**

**input**

```
3
4 2
4 1 3 2
3 5
-2 -7 -1
10 2
-6 -1 -2 4 -6 -1 -4 4 -5 -4
```

**output**

```
10 12 14 16 18
0 4 4 5
4 6 6 7 7 7 7 8 8 8 8
```

### Note

In the first testcase, it doesn't matter which elements you add $x$ to. The subarray with the maximum sum will always be the entire array. If you increase $k$ elements by $x$, $k \cdot x$ will be added to the sum.

In the second testcase:

- For $k = 0$, the empty subarray is the best option.
- For $k = 1$, it's optimal to increase the element at position $3$. The best sum becomes $-1 + 5 = 4$ for a subarray $[3, 3]$.

- For $k = 2$, it's optimal to increase the element at position $3$ and any other element. The best sum is still $4$ for a subarray $[3, 3]$.
- For $k = 3$, you have to increase all elements. The best sum becomes $(-2 + 5) + (-7 + 5) + (-1 + 5) = 5$ for a subarray $[1, 3]$.

## D. Cross Coloring

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

There is a sheet of paper that can be represented with a grid of size $n \times m$: $n$ rows and $m$ columns of cells. All cells are colored in white initially.

$q$ operations have been applied to the sheet. The $i$-th of them can be described as follows:

- $x_i$ $y_i$ — choose one of $k$ non-white colors and color the entire row $x_i$ and the entire column $y_i$ in it. The new color is applied to each cell, regardless of whether the cell was colored before the operation.

The sheet after applying all $q$ operations is called a coloring. Two colorings are different if there exists at least one cell that is colored in different colors.

How many different colorings are there? Print the number modulo $998\,244\,353$.

### Input

The first line contains a single integer $t$ ($1 \le t \le 10^4$) — the number of testcases.

The first line of the testcase contains four integers $n, m, k$ and $q$ ($1 \le n, m, k, q \le 2 \cdot 10^5$) — the size of the sheet, the number of non-white colors and the number of operations.

The $i$-th of the following $q$ lines contains a description of the $i$-th operation — two integers $x_i$ and $y_i$ ($1 \le x_i \le n$; $1 \le y_i \le m$) — the row and the column the operation is applied to.

The sum of $q$ over all testcases doesn't exceed $2 \cdot 10^5$.

### Output

For each testcase, print a single integer — the number of different colorings modulo $998\,244\,353$.

### Example

**input**

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

**output**

```
3
4
```

## E. Expand the Path

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

Consider a grid of size $n \times n$. The rows are numbered top to bottom from $1$ to $n$, the columns are numbered left to right from $1$ to $n$.

The robot is positioned in a cell $(1, 1)$. It can perform two types of moves:

- D — move one cell down;
- R — move one cell right.

The robot is not allowed to move outside the grid.

You are given a sequence of moves $s$ — the initial path of the robot. This path doesn't lead the robot outside the grid.

You are allowed to perform an arbitrary number of modifications to it (possibly, zero). With one modification, you can duplicate one move in the sequence. That is, replace a single occurrence of D with DD or a single occurrence of R with RR.

Count the number of cells such that there exists at least one sequence of modifications that the robot visits this cell on the modified path and doesn't move outside the grid.

## Input

The first line contains a single integer $t$ ($1 \le t \le 10^4$) — the number of testcases.

The first line of each testcase contains the single integer $n$ ($2 \le n \le 10^8$) — the number of rows and columns in the grid.

The second line of each testcase contains a non-empty string $s$, consisting only of characters D and R, — the initial path of the robot. This path doesn't lead the robot outside the grid.

The total length of strings $s$ over all testcases doesn't exceed $2 \cdot 10^5$.

## Output

For each testcase, print a single integer — the number of cells such that there exists at least one sequence of modifications that the robot visits this cell on the modified path and doesn't move outside the grid.

### Example

**input**

```
3
4
RD
5
DRDRDRDR
3
D
```

**output**

```
13
9
3
```

## Note

In the first testcase, it's enough to consider the following modified paths:

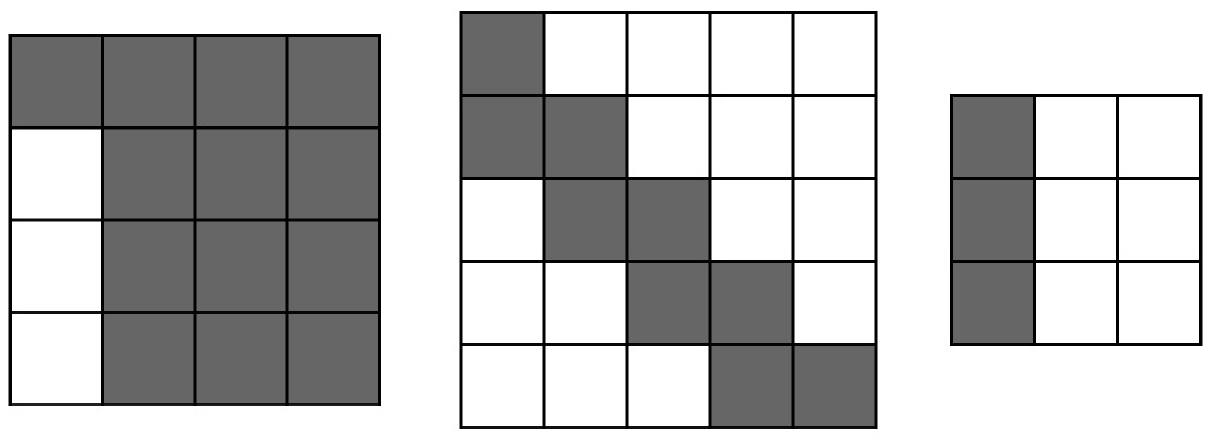
- RD $\to$ RRD $\to$ RRRD $\to$ RRRDD $\to$ RRRDDD — this path visits cells $(1, 1)$, $(1, 2)$, $(1, 3)$, $(1, 4)$, $(2, 4)$, $(3, 4)$ and $(4, 4)$;
- RD $\to$ RRD $\to$ RRDD $\to$ RRDDD — this path visits cells $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 3)$, $(3, 3)$ and $(4, 3)$;
- RD $\to$ RDD $\to$ RDDD — this path visits cells $(1, 1)$, $(1, 2)$, $(2, 2)$, $(3, 2)$ and $(4, 2)$.

Thus, the cells that are visited on at least one modified path are: $(1, 1)$, $(1, 2)$, $(1, 3)$, $(1, 4)$, $(2, 2)$, $(2, 3)$, $(2, 4)$, $(3, 2)$, $(3, 3)$, $(3, 4)$, $(4, 2)$, $(4, 3)$ and $(4, 4)$.

In the second testcase, there is no way to modify the sequence without moving the robot outside the grid. So the only visited cells are the ones that are visited on the path DRDRDRDR.

In the third testcase, the cells that are visited on at least one modified path are: $(1, 1)$, $(2, 1)$ and $(3, 1)$.

Here are the cells for all testcases:



# F. Basis

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

For an array of integers $a$, let's define $|a|$ as the number of elements in it.

Let's denote two functions:

- $F(a, k)$ is a function that takes an array of integers $a$ and a positive integer $k$. The result of this function is the array containing $|a|$ first elements of the array that you get by replacing each element of $a$ with exactly $k$ copies of that element.
  For example, $F([2, 2, 1, 3, 5, 6, 8], 2)$ is calculated as follows: first, you replace each element of the array with $2$ copies of it, so you obtain $[2, 2, 2, 2, 1, 1, 3, 3, 5, 5, 6, 6, 8, 8]$. Then, you take the first $7$ elements of the array you obtained, so the result of the function is $[2, 2, 2, 2, 1, 1, 3]$.
- $G(a, x, y)$ is a function that takes an array of integers $a$ and two **different** integers $x$ and $y$. The result of this function is the array $a$ with every element equal to $x$ replaced by $y$, and every element equal to $y$ replaced by $x$.
  For example, $G([1, 1, 2, 3, 5], 3, 1) = [3, 3, 2, 1, 5]$.

An array $a$ is a **parent** of the array $b$ if:

- either there exists a positive integer $k$ such that $F(a, k) = b$;
- or there exist two different integers $x$ and $y$ such that $G(a, x, y) = b$.

An array $a$ is an **ancestor** of the array $b$ if there exists a finite sequence of arrays $c_0, c_1, \dots, c_m$ ($m \ge 0$) such that $c_0$ is $a$, $c_m$ is $b$, and for every $i \in [1, m]$, $c_{i-1}$ is a parent of $c_i$.

*And now, the problem itself.*

You are given two integers $n$ and $k$. Your goal is to construct a sequence of arrays $s_1, s_2, \dots, s_m$ in such a way that:

- every array $s_i$ contains exactly $n$ elements, and all elements are integers from $1$ to $k$;
- for every array $a$ consisting of exactly $n$ integers from $1$ to $k$, the sequence contains at least one array $s_i$ such that $s_i$ is an ancestor of $a$.

Print the minimum number of arrays in such sequence.

## Input

The only line contains two integers $n$ and $k$ ($1 \le n, k \le 2 \cdot 10^5$).

## Output

Print one integer — the minimum number of elements in a sequence of arrays meeting the constraints. Since the answer can be large, output it modulo $998244353$.

### Examples

**input**
```
3 2
```
**output**
```
2
```

**input**
```
4 10
```
**output**
```
12
```

**input**
```
13 37
```
**output**
```
27643508
```

**input**
```
1337 42
```
**output**
```
211887828
```

**input**
```
198756 123456
```
**output**
```
159489391
```

**input**
```
123456 198756
```
**output**
```
460526614
```

## Note

Let's analyze the first example.

One of the possible answers for the first example is the sequence $[[2, 1, 2], [1, 2, 2]]$. Every array of size $3$ consisting of elements from $1$ to $2$ has an ancestor in this sequence:

- for the array $[1, 1, 1]$, the ancestor is $[1, 2, 2]$: $F([1, 2, 2], 13) = [1, 1, 1]$;
- for the array $[1, 1, 2]$, the ancestor is $[1, 2, 2]$: $F([1, 2, 2], 2) = [1, 1, 2]$;
- for the array $[1, 2, 1]$, the ancestor is $[2, 1, 2]$: $G([2, 1, 2], 1, 2) = [1, 2, 1]$;
- for the array $[1, 2, 2]$, the ancestor is $[1, 2, 2]$;
- for the array $[2, 1, 1]$, the ancestor is $[1, 2, 2]$: $G([1, 2, 2], 1, 2) = [2, 1, 1]$;
- for the array $[2, 1, 2]$, the ancestor is $[2, 1, 2]$;
- for the array $[2, 2, 1]$, the ancestor is $[2, 1, 2]$: $F([2, 1, 2], 2) = [2, 2, 1]$;
- for the array $[2, 2, 2]$, the ancestor is $[1, 2, 2]$: $G(F([1, 2, 2], 4), 1, 2) = G([1, 1, 1], 1, 2) = [2, 2, 2]$.