Moving Intervals

There are **C** cakes in a row, numbered from 1 to **C**. There are **N** children, each of whom have selected a consecutive set of cakes to eat. That is, Child i has decided to eat all the cakes from $\mathbf{S_i}$ to $\mathbf{E_i}$, end points inclusive. If there is a cake which appears in some two childrens' set, then they will fight because both of them want to eat that cake, and you don't want that to happen.

You will be given an integer **K** which will be either 0 or 1. If **K** is 0, then you should find out if some two children will fight. Print "Good" if no one fights, and "Bad" if someone fights.

If **K** is 1, then you can persuade at most one child to change his decision to some other set of cakes. But the number of cakes that he eats must be the same. That is, if Child i had initially decided that he wants to eat the cakes from $\mathbf{S_i}$ to $\mathbf{E_i}$, then you could persuade the child to instead eat the cakes from X to Y instead, for any valid X and Y (ie. $1 \le X \le Y \le \mathbf{C}$), provided that the number of cakes is the same (ie. $\mathbf{E_i} - \mathbf{S_i} + 1 = Y - X + 1$). If after persuading at most 1 Child to change his decision, no fights happen, then print "Good". But if no matter what you do, someone will fight, then print "Bad".

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of each test case follows.

The first line of each test case contains three integers **C**, **N** and **K** denoting the number of cakes, number of children and **K**, respectively.

The i-th of the next **N** lines contains two space separated integers $\mathbf{S_i}$ and $\mathbf{E_i}$ which denotes the initial decision of Child i. That is, Child i wants to eat from cake $\mathbf{S_i}$ to cake $\mathbf{E_i}$.

Output

For each test case, output a single line containing "Good" or "Bad".

Constraints

- $1 \le \mathbf{T} \le 10$
- $1 \le \mathbf{C} \le 10^9$
- $1 \le \mathbf{N} \le 10^5$
- $0 \le \mathbf{K} \le 1$
- $1 \le \mathbf{S_i}, \mathbf{E_i} \le \mathbf{C}$

Subtasks:

- Subtask #1 (15 points):
  - $K = 0$
  - $1 \le \mathbf{N} \le 1000$
  - $\mathbf{E_i} \le \mathbf{C}/2$, for all $1 \le i \le \mathbf{N}$
- Subtask #2 (10 points):
  - $1 \le \mathbf{N} \le 1000$
  - $\mathbf{E_i} \le \mathbf{C}/2$, for all $1 \le i \le \mathbf{N}$
- Subtask #3 (25 points):
  - $1 \le \mathbf{N} \le 1000$
- Subtask #4 (50 points): Original constraints.

Example

```
Input:
3
5 2 0
2 2
3 5
5 2 1
2 2
2 5
5 2 1
2 3
2 5

Output:
Good
Good
Bad
```

Explanation

Test case 1: Child 1 wants to eat the second cake, and Child 2 wants to eat Cakes 3, 4 and 5. So there is no fight, and the answer is "Good".

Test case 2: Child 1 wants to eat Cake 2, and Child 2 wants to eat Cakes 2, 3, 4 and 5. Both of them want to eat Cake 2, and hence it could lead to a fight. But because **K** = 1, we can persuade one of the children to change their decision. For instance, we could persuade Child 1 to change his decision from [2, 2] to [1, 1]. After this, there is no fight, and the hence answer is "Good".

Test case 3: Child 1 wants to eat Cake 2 and Cake 3, and Child 2 wants to eat Cakes 2, 3, 4 and 5. Both of them want to eat Cakes 2 and 3, and hence it could lead to a fight. And because **K** = 1, we can persuade one of the children to change their decision. For instance, we could persuade Child 1 to change his decision from [2, 3] to [1, 2]. But even after this, both of them want to eat Cake 2. You can verify that no matter how we persuade at most 1 child, they will end up fighting. Hence the answer is "Bad".

Resource Constraints

Time limit: 1 sec

Memory: 1 GB

String Importance

You are given a string of **N** characters, ($\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_3}$, ... , $\mathbf{S_N}$), in which each character is 'X', 'Y' or 'Z'. A substring is a consecutive portion of the string. That is, a substring will be of the form ($\mathbf{S_i}$, $\mathbf{S_{i+1}}$, ... , $\mathbf{S_j}$), for some i and j. A substring is said to be *Good* if it starts with an 'X', ends with a 'Z' and has a length which is a multiple of 3.

The *Importance* of a particular substring ($\mathbf{S_i}$, $\mathbf{S_{i+1}}$, ... , $\mathbf{S_j}$), is the number of *Good* substrings that it intersects with. That is, it is the number of x and y, such that ($\mathbf{S_x}$, $\mathbf{S_{x+1}}$, ... , $\mathbf{S_y}$) forms a *Good* substring, and there is some z such that i ≤ z ≤ j and x ≤ z ≤ y. This signifies that there is at least a single index z, which should be common.

Given an integer **K**, you need to find a substring of length exactly **K** which has the minimum *Importance*, and print the minimum *Importance*.

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of each test case follows.

The first line of each test case contains two integers **N** and **K** denoting the number of characters in the input string and the length of the substring needed.

The second line of each test case contains **N** space separated characters denoting the input string.

Output

For each test case, output a single line containing a single integer, which should be the minimum *Importance* of a **K** length substring.

Constraints

- 1 ≤ **T** ≤ 10
- 1 ≤ **K** ≤ **N** ≤ 10^5

Subtasks:

- Subtask #1 (10 points): 1 ≤ **K** ≤ **N** ≤ 100
- Subtask #2 (15 points): 1 ≤ **K** ≤ **N** ≤ 1000
- Subtask #3 (35 points): **K** = 1
- Subtask #4 (40 points): Original constraints.

Example

```
Input:
2
9 3
X Y Z Z Y X X Y Z
4 2
Y X Y Z

Output:
1
1
```

Explanation

Test case 1:

The input string is (X, Y, Z, Z, Y, X, X, Y, Z). There are totally 3 *Good* substrings, which are underlined:

- (**X**, **Y**, **Z**, Z, Y, X, X, Y, Z)
- (X, Y, Z, Z, Y, X, **X**, **Y**, **Z**)
- (**X**, **Y**, **Z**, **Z**, **Y**, **X**, **X**, **Y**, **Z**)

We now need to find a substring of length 3 which intersects with the minimum number of *Good* substrings. In this example, there is an unique such substring: (X, Y, Z, **Z**, **Y**, **X**, X, Y, Z). Its *Importance* is 1 because it intersects with only the third *Good* substring. We cannot do better. Hence the answer is 1.

Test case 2:

The input string is (Y, X, Y, Z). There is only one *Good* substring: (Y, **X**, **Y**, **Z**). But every substring of length 2 intersects with this *Good* substring, and hence the *Importance* of every substring of length 2 is 1. Therefore the minimum is also 1, and that is the answer.

Resource Constraints

Time limit: 1 sec

Memory: 1 GB