Codeforces Round #668 (Div. 1)

A. Balanced Bitstring

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

A bitstring is a string consisting only of the characters `0` and `1`. A bitstring is called k-**balanced** if every substring of size k of this bitstring has an equal amount of `0` and `1` characters ($\frac{k}{2}$ of each).

You are given an integer k and a string s which is composed only of characters `0`, `1`, and `?`. You need to determine whether you can make a k-balanced bitstring by replacing every `?` characters in s with either `0` or `1`.

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains two integers n and k ($2 \le k \le n \le 3 \cdot 10^5$, k is even) — the length of the string and the parameter for a balanced bitstring.

The next line contains the string s ($|s| = n$). It is given that s consists of only `0`, `1`, and `?`.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print YES if we can replace every `?` in s with `0` or `1` such that the resulting bitstring is k-balanced, or `NO` if it is not possible.

Example

input
9 6 4 100110 3 2 1?1 3 2 1?0 4 4 ???? 7 4 1?0??1? 10 10 11??11??11 4 2 1??1 4 4 ?0?0 6 2 ????00
output
YES YES NO YES YES NO NO YES NO

Note

For the first test case, the string is already a 4-balanced bitstring.

For the second test case, the string can be transformed into `101`.

For the fourth test case, the string can be transformed into `0110`.

For the fifth test case, the string can be transformed into `1100110`.

B. Tree Tag

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

Alice and Bob are playing a fun game of tree tag.

The game is played on a tree of n vertices numbered from 1 to n. Recall that a tree on n vertices is an undirected, connected graph with $n-1$ edges.

Initially, Alice is located at vertex a, and Bob at vertex b. They take turns alternately, and Alice makes the first move. In a move, Alice can jump to a vertex with distance **at most** da from the current vertex. And in a move, Bob can jump to a vertex with distance **at most** db from the current vertex. The distance between two vertices is defined as the number of edges on the unique simple path between them. In particular, either player is allowed to stay at the same vertex in a move. Note that when performing a move, a player only occupies the starting and ending vertices of their move, not the vertices between them.

If after at most 10^{100} moves, Alice and Bob occupy the same vertex, then Alice is declared the winner. Otherwise, Bob wins.

Determine the winner if both players play optimally.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains five integers n, a, b, da, db ($2 \le n \le 10^5$, $1 \le a, b \le n$, $a \ne b$, $1 \le da, db \le n-1$) — the number of vertices, Alice's vertex, Bob's vertex, Alice's maximum jumping distance, and Bob's maximum jumping distance, respectively.

The following $n-1$ lines describe the edges of the tree. The i-th of these lines contains two integers u, v ($1 \le u, v \le n, u \ne v$), denoting an edge between vertices u and v. It is guaranteed that these edges form a tree structure.

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output

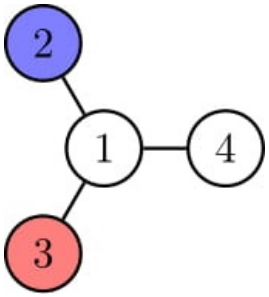
For each test case, output a single line containing the winner of the game: "Alice" or "Bob".

Example

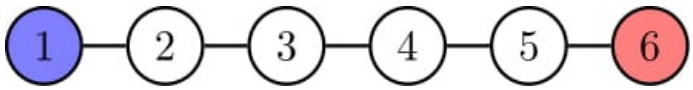
input
4 4 3 2 1 2 1 2 1 3 1 4 6 6 1 2 5 1 2 6 5 2 3 3 4 4 5 9 3 9 2 5 1 2 1 6 1 9 1 3 9 5 7 9 4 8 4 3 11 8 11 3 3 1 2 11 9 4 9 6 5 2 10 3 2 5 9 8 3 7 4 7 10
output
Alice Bob Alice Alice

Note

In the first test case, Alice can win by moving to vertex 1. Then wherever Bob moves next, Alice will be able to move to the same vertex on the next move.



In the second test case, Bob has the following strategy to win. Wherever Alice moves, Bob will always move to whichever of the two vertices 1 or 6 is farthest from Alice.



C. Fixed Point Removal

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let $a_1, \ldots, a_n$ be an array of n positive integers. In one operation, you can choose an index i such that $a_i = i$, and remove a_i from the array (after the removal, the remaining parts are concatenated).

The weight of a is defined as the maximum number of elements you can remove.

You must answer q independent queries (x, y): after replacing the x first elements of a and the y last elements of a by $n+1$ (making them impossible to remove), what would be the weight of a?

Input

The first line contains two integers n and q ($1 \leq n, q \leq 3 \cdot 10^5$) — the length of the array and the number of queries.

The second line contains n integers a_1, a_2, ..., a_n ($1 \leq a_i \leq n$) — elements of the array.

The i-th of the next q lines contains two integers x and y ($x, y \geq 0$ and $x + y < n$).

Output

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

Examples

input
13 5 2 2 3 9 5 4 6 5 7 8 3 11 13 3 1 0 0 2 4 5 0 0 12
output
5 11 6 1 0

input
5 2 1 4 1 2 4 0 0 1 0
output
2 0

Note

Explanation of the first query:

After making first $x = 3$ and last $y = 1$ elements impossible to remove, a becomes $[\times, \times, \times, 9, 5, 4, 6, 5, 7, 8, 3, 11, \times]$ (we represent 14 as $\times$ for clarity).

Here is a strategy that removes 5 elements (the element removed is colored in red):

- $[\times, \times, \times, 9, \color{red}{5}, 4, 6, 5, 7, 8, 3, 11, \times]$
- $[\times, \times, \times, 9, 4, 6, 5, 7, 8, 3, \color{red}{11}, \times]$

- $[\times, \times, \times, 9, 4, 6, 5, 7, 8, 3, \times]$
- $[\times, \times, \times, 9, 4, 5, 7, 8, 3, \times]$
- $[\times, \times, \times, 9, 4, 5, 7, 3, \times]$
- $[\times, \times, \times, 9, 4, 5, 3, \times]$ (final state)

It is impossible to remove more than 5 elements, hence the weight is 5.

D. Game of Pairs

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

Consider a fixed positive integer n. Two players, First and Second play a game as follows:

1. First considers the $2n$ numbers $1, 2, \ldots, 2n$, and partitions them as he wants into n disjoint pairs.
2. Then, Second chooses exactly one element from each of the pairs that First created (he chooses elements he wants).

To determine the winner of the game, we compute the sum of the numbers chosen by Second. If the sum of all these numbers is a multiple of $2n$, then Second wins. Otherwise, First wins.

You are given the integer n. Your task is to decide which player you wish to play as and win the game.

Interaction

The interaction begins by reading the integer n ($1 \le n \le 5 \cdot 10^5$).

After reading, print a single line containing either First or Second, denoting who you want to play as. The interaction then varies depending on who you chose to play as.

If you chose to play as First, print a single line containing $2n$ integers $p_1, p_2, \ldots, p_{2n}$, denoting that the number i belongs to the p_i-th pair for $1 \le i \le 2n$. Thus, $1 \le p_i \le n$, and every number between 1 and n inclusive should appear exactly twice.

If you chose to play as Second, the interactor will print $2n$ integers $p_1, p_2, \ldots, p_{2n}$, denoting that the number i belongs to the p_i-th pair. As a response, print n integers $a_1, a_2, \ldots, a_n$ in a single line. These should contain exactly one number from each pair.

Regardless of who you chose to play as the interactor will finish by printing a single integer: 0 if your answer for the test case is correct (that is, you are playing as First and it cannot choose adequate numbers from your pairs, or you are playing as Second and your chosen numbers add up to a multiple of $2n$), or -1 if it is incorrect. In particular, the interactor will not print the chosen numbers if you choose to play First and lose. In either case, your program should terminate immediately after reading this number.

If at any point you make an invalid interaction, the interactor will print -1 and finish the interaction. You will receive a **Wrong Answer** verdict. Make sure to terminate immediately to avoid getting other verdicts.

After printing something 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

To hack, use the following format:

The first line contains an integer n ($1 \le n \le 5 \cdot 10^5$).

The second line contains $2n$ integers $p_1, p_2, \ldots, p_{2n}$, denoting that the number i belongs to the p_i-th pair if the solution being hacked chooses to play as Second. If the solution being hacked chooses to play as First, those pairs don't matter but the $p_1, p_2, \ldots, p_{2n}$ must still form a valid partition of $1, 2, \ldots, 2n$ into n disjoint pairs.

Examples

input
2 1 1 2 2 0
output
Second

1 3

input
2   0
output
First 2 1 2 1

Note

In the first sample, $n = 2$, and you decide to play as Second. The judge chooses the pairs $(1, 2)$ and $(3, 4)$, and you reply with the numbers 1 and 3. This is a valid choice since it contains exactly one number from each pair, and the sum $1 + 3 = 4$ is divisible by 4.

In the second sample, $n = 2$ again, and you play as First. You choose the pairs $(2, 4)$ and $(1, 3)$. The judge fails to choose a number from each pair such that their sum is divisible by 4, so the answer is correct.

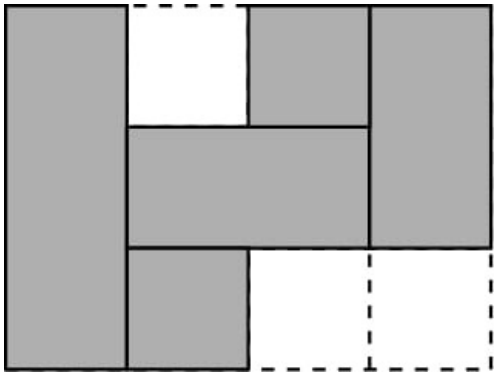
Note that the sample tests are just for illustration of the interaction protocol, and don't necessarily correspond to the behavior of the real interactor.

E. Bricks

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A brick is defined as a rectangle with integer side lengths with either width 1 or height 1 (or both).

There is an $n \times m$ grid, and each cell is colored either black or white. A tiling is a way to place bricks onto the grid such that each black cell is covered by exactly one brick, and each white cell is not covered by any brick. In other words, bricks are placed on black cells only, cover all black cells, and **no two bricks overlap**.



An example tiling of the first test case using 5 bricks. It is possible to do better, using only 4 bricks.

What is the minimum number of bricks required to make a valid tiling?

Input

The first line contains two integers n, m ($1 \leq n, m \leq 200$) — the number of rows and columns, respectively.

The next n lines describe the grid. The i-th of these lines contains a string of length m, where the j-th character denotes the color of the cell in row i, column j. A black cell is given by "#", and a white cell is given by "`.`".

It is guaranteed that there is at least one black cell.

Output

Output a single integer, the minimum number of bricks required.

Examples

input
3 4 #.## #### ##..
output
4

input
6 6 ######

```
##....
######
##...#
##...#
######
```

output

```
6
```

input

```
10 8
####..##
#..#.##.
#..#.###
####.#.#
....####
.###.###
###.#..#
########
###..###
.##.###.
```

output

```
18
```

Note

The first test case can be tiled with 4 bricks placed vertically.

The third test case can be tiled with 18 bricks like this: