Problem A. Binary String Minimizing

Time limit 1000 ms
Mem limit 262144 kB

You are given a binary string of length n (i. e. a string consisting of n characters '`0`' and '`1`').

In one move you can swap two adjacent characters of the string. What is the lexicographically minimum possible string you can obtain from the given one if you can perform **no more** than k moves? It is possible that you do not perform any moves at all.

Note that you can swap the same pair of adjacent characters with indices i and $i + 1$ arbitrary (possibly, zero) number of times. Each such swap is considered a separate move.

You have to answer q independent test cases.

Input

The first line of the input contains one integer q ($1 \le q \le 10^4$) — the number of test cases.

The first line of the test case contains two integers n and k ($1 \le n \le 10^6, 1 \le k \le n^2$) — the length of the string and the number of moves you can perform.

The second line of the test case contains one string consisting of n characters '`0`' and '`1`'.

It is guaranteed that the sum of n over all test cases does not exceed 10^6 ($\sum n \le 10^6$).

Output

For each test case, print the answer on it: the lexicographically minimum possible string of length n you can obtain from the given one if you can perform **no more** than k moves.

Examples

Input	Output
3 8 5 11011010 7 9 1111100 7 11 1111100	01011110 0101111 0011111

Note

In the first example, you can change the string as follows: $1\underline{10}11010 \rightarrow \underline{10}111010 \rightarrow 0111\underline{10}10 \rightarrow 011\underline{10}110 \rightarrow 01\underline{10}1110 \rightarrow 01011110$.

In the third example, there are enough operations to make the string sorted.

Problem B. Find and Replace

Time limit 1000 ms
Mem limit 262144 kB

You are given a string s consisting of lowercase Latin characters. In an operation, you can take a character and replace **all** occurrences of this character with `0` or replace **all** occurrences of this character with `1`.

Is it possible to perform some number of moves so that the resulting string is an alternating binary string[†]?

For example, consider the string **`abacaba`**. You can perform the following moves:

- Replace `a` with `0`. Now the string is `0b0c0b0`.
- Replace `b` with `1`. Now the string is `010c010`.
- Replace `c` with `1`. Now the string is `01011010`. This is an alternating binary string.

[†]An *alternating binary string* is a string of `0`s and `1`s such that no two adjacent bits are equal. For example, `01010101`, `101`, `1` are alternating binary strings, but `0110`, `0a0a0`, `10100` are not.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 2000$) — the length of the string s.

The second line of each test case contains a string consisting of n lowercase Latin characters — the string s.

Output

For each test case, output "`YES`" (without quotes) if you can make the string into an alternating binary string, and "`NO`" (without quotes) otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

Examples

Input	Output
8 7 abacaba 2 aa 1 y 4 bkpt 6 ninfia 6 banana 10 codeforces 8 testcase	YES NO YES YES NO YES NO NO

Note

The first test case is explained in the statement.

In the second test case, the only possible binary strings you can make are 00 and 11, neither of which are alternating.

In the third test case, you can make 1, which is an alternating binary string.

Problem C. Platforms Jumping

Time limit 1000 ms
Mem limit 262144 kB

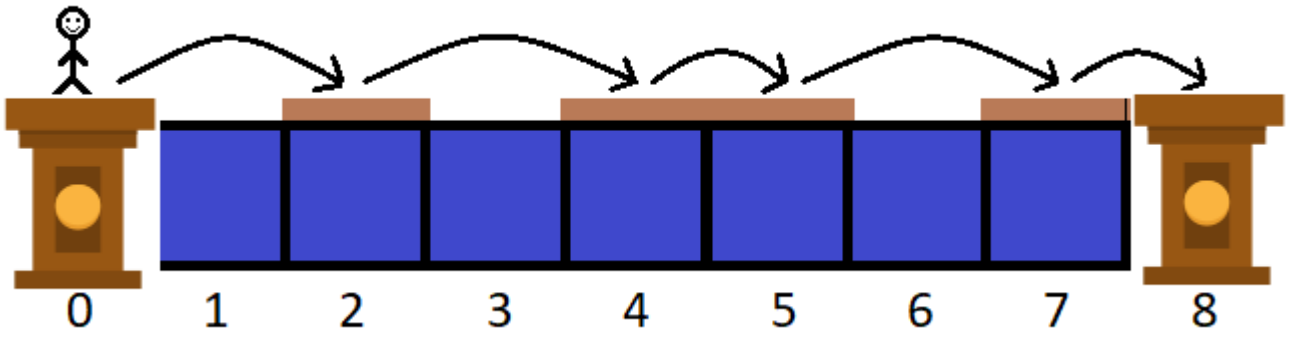
There is a river of width n. The left bank of the river is cell 0 and the right bank is cell $n+1$ (more formally, the river can be represented as a sequence of $n+2$ cells numbered from 0 to $n+1$). There are also m wooden platforms on a river, the i-th platform has length c_i (so the i-th platform takes c_i consecutive cells of the river). It is guaranteed that the sum of lengths of platforms does not exceed n.

You are standing at 0 and want to reach $n+1$ somehow. If you are standing at the position x, you can jump to any position in the range $[x+1; x+d]$. **However** you don't really like the water so you can jump only to such cells that belong to some wooden platform. For example, if $d=1$, you can jump only to the next position (if it belongs to the wooden platform). **You can assume that cells** 0 **and** $n+1$ **belong to wooden platforms**.

You want to know if it is possible to reach $n+1$ from 0 if you can move any platform to the left or to the right arbitrary number of times (possibly, zero) as long as they do not intersect each other (but two platforms can touch each other). It also means that you cannot change the relative order of platforms.

Note that you should move platforms until you start jumping (in other words, you first move the platforms and then start jumping).

For example, if $n=7$, $m=3$, $d=2$ and $c=[1, 2, 1]$, then one of the ways to reach 8 from 0 is follow:



The first example: $n=7$.

Input

The first line of the input contains three integers n, m and d ($1 \le n, m, d \le 1000, m \le n$) — the width of the river, the number of platforms and the maximum distance of your jump, correspondingly.

The second line of the input contains m integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le n, \sum\limits_{i=1}^{m} c_i \le n$), where c_i is the length of the i-th platform.

Output

If it is impossible to reach $n + 1$ from 0, print `NO` in the first line. Otherwise, print `YES` in the first line and the array a of length n in the second line — the sequence of river cells (excluding cell 0 and cell $n + 1$).

If the cell i does not belong to any platform, a_i should be 0. Otherwise, it should be equal to the index of the platform (1-indexed, platforms are numbered from 1 to m in order of input) to which the cell i belongs.

Note that all a_i equal to 1 should form a contiguous subsegment of the array a of length c_1, all a_i equal to 2 should form a contiguous subsegment of the array a of length c_2, ..., all a_i equal to m should form a contiguous subsegment of the array a of length c_m. The leftmost position of 2 in a should be greater than the rightmost position of 1, the leftmost position of 3 in a should be greater than the rightmost position of 2, ..., the leftmost position of m in a should be greater than the rightmost position of $m - 1$.

See example outputs for better understanding.

Examples

Input	Output
7 3 2 1 2 1	YES 0 1 0 2 2 0 3

Input	Output
10 1 11 1	YES 0 0 0 0 0 0 0 0 0 1

Input	Output
10 1 5 2	YES 0 0 0 0 1 1 0 0 0 0

Note

Consider the first example: the answer is $[0, 1, 0, 2, 2, 0, 3]$. The sequence of jumps you perform is $0 \to 2 \to 4 \to 5 \to 7 \to 8$.

Consider the second example: it does not matter how to place the platform because you always can jump from 0 to 11.

Consider the third example: the answer is $[0, 0, 0, 0, 1, 1, 0, 0, 0, 0]$. The sequence of jumps you perform is $0 \to 5 \to 6 \to 11$.

Problem D. Bits

Time limit	1000 ms
Mem limit	262144 kB
Input file	stdin
Output file	stdout

Let's denote as $\text{popcount}(x)$ the number of bits set ('1' bits) in the binary representation of the non-negative integer x.

You are given multiple queries consisting of pairs of integers l and r. For each query, find the x, such that $l \le x \le r$, and $\text{popcount}(x)$ is maximum possible. If there are multiple such numbers find the smallest of them.

Input

The first line contains integer n — the number of queries ($1 \le n \le 10000$).

Each of the following n lines contain two integers l_i, r_i — the arguments for the corresponding query ($0 \le l_i \le r_i \le 10^{18}$).

Output

For each query print the answer in a separate line.

Examples

Input	Output
3 1 2 2 4 1 10	1 3 7

Note

The binary representations of numbers from 1 to 10 are listed below:

$1_{10} = 1_2$

$2_{10} = 10_2$

$3_{10} = 11_2$

$4_{10} = 100_2$

$5_{10} = 101_2$

$6_{10} = 110_2$

$7_{10} = 111_2$

$8_{10} = 1000_2$

$9_{10} = 1001_2$

$10_{10} = 1010_2$

Problem E. 01 Tree

Time limit 1000 ms
Mem limit 262144 kB

There is an edge-weighted complete binary tree with n leaves. A complete binary tree is defined as a tree where every non-leaf vertex has exactly 2 children. For each non-leaf vertex, we label one of its children as the left child and the other as the right child.

The binary tree has a very strange property. For every non-leaf vertex, one of the edges to its children has weight 0 while the other edge has weight 1. Note that the edge with weight 0 can be connected to either its left or right child.

You forgot what the tree looks like, but luckily, you still remember some information about the leaves in the form of an array a of size n. For each i from 1 to n, a_i represents the distance[†] from the root to the i-th leaf in dfs order[‡]. Determine whether there exists a complete binary tree which satisfies array a. Note that you **do not** need to reconstruct the tree.

[†] The distance from vertex u to vertex v is defined as the sum of weights of the edges on the path from vertex u to vertex v.

[‡] The dfs order of the leaves is found by calling the following `dfs` function on the root of the binary tree.

```
dfs_order = []

function dfs(v):
    if v is leaf:
        append v to the back of dfs_order
    else:
        dfs(left child of v)
        dfs(right child of v)

dfs(root)
```

Input

Each test contains multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the size of array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n - 1$) — the distance from the root to the i-th leaf.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if there exists a complete binary tree which satisfies array a and "NO" otherwise.

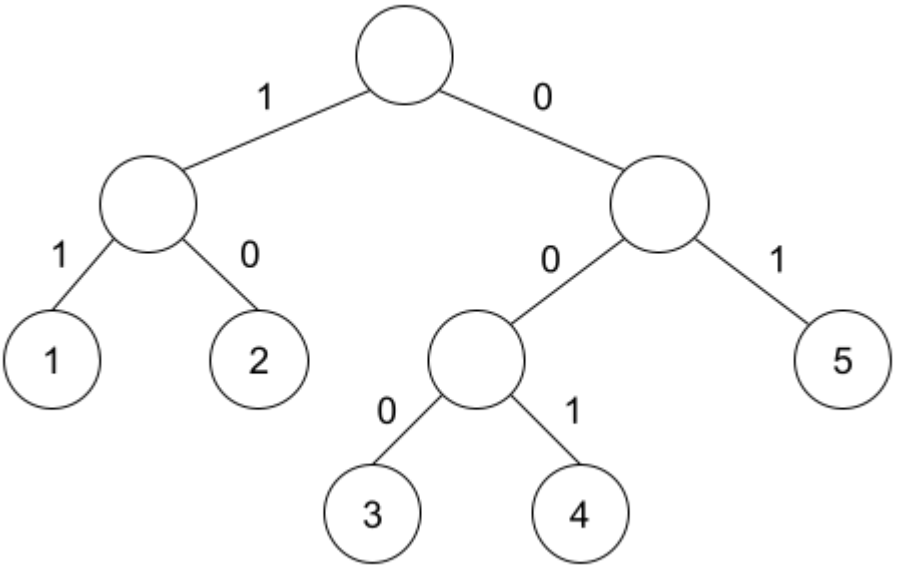
You may print each letter in any case (for example, "YES", "Yes", "yes", "yEs" will all be recognized as a positive answer).

Examples

Input	Output
2 5 2 1 0 1 1 5 1 0 2 1 3	YES NO

Note

In the first test case, the following tree satisfies the array.



In the second test case, it can be proven that there is no complete binary tree that satisfies the array.

Problem F. Vabank

Time limit 2000 ms
Mem limit 262144 kB

Gustaw is the chief bank manager in a huge bank. He has unlimited access to the database system of the bank, in a few clicks he can move any amount of money from the bank's reserves to his own private account. However, the bank uses some fancy AI fraud detection system that makes stealing more difficult.

Gustaw knows that the anti-fraud system just detects any operation that exceeds some fixed limit M euros and these operations are checked manually by a number of clerks. Thus, any fraud operation exceeding this limit is detected, while any smaller operation gets unnoticed.

Gustaw doesn't know the limit M and wants to find it out. In one operation, he can choose some integer X and try to move X euros from the bank's reserves to his own account. Then, the following happens.

- If $X \leq M$, the operation is unnoticed and Gustaw's account balance raises by X euros.
- Otherwise, if $X > M$, the fraud is detected and cancelled. Moreover, Gustaw has to pay X euros from his own account as a fine. If he has less than X euros on the account, he is fired and taken to the police.

Initially Gustaw has 1 euro on his account. Help him find the exact value of M in no more than 105 operations without getting him fired.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$).

For each test case, there is no input prior to your first query, but you can be sure that M is integer, and $0 \leq M \leq 10^{14}$.

Output

For each test case, when you know the exact value of M, print a single line with format "`! M`". After that your program should proceed to the next test case or terminate, if it is the last one.

Interaction

When you want to make an operation, print a single line with format "`? X`", denoting that you try to move X euros ($1 \le X \le 10^{14}$). As a response, read a single line that can take the following values:

- "`Lucky!`", if $X \le M$. Your balance raises by X.
- "`Fraudster!`", if $X > M$. Your balance decreases by X.
- "`Fired!`", if $X > M$, but you can't pay X euros of fine. In this case your program must terminate immediately. You also get this response if you exceed the number of queries.

You can make at most 105 such queries in each test case.

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.

Hacks

To make a hack, prepare an input in the following format.

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

Each test case is described with a line containing a single integer M ($0 \le M \le 10^{14}$).

Examples

Input	Output
1 Lucky! Lucky! Lucky! Lucky! Lucky! Fraudster!	? 1 ? 2 ? 3 ? 4 ? 5 ? 6 ! 5

Note

In the example $M = 5$, so the operation with $X = 6$ is detected. At this moment, Gustaw's balance is 16 euros, so he just pays fine.