Educational Codeforces Round 101 (Rated for Div. 2)

A. Regular Bracket Sequence

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

A bracket sequence is called regular if it is possible to obtain correct arithmetic expression by inserting characters + and 1 into this sequence. For example, sequences `(())()`, `()` and `(()(()))` are regular, while `)(`, `(()` and `(()))(` are not. Let's call a regular bracket sequence "RBS".

You are given a sequence s of n characters `(`, `)`, and/or `?`. **There is exactly one character `(` and exactly one character `)`** in this sequence.

You have to replace every character `?` with either `)` or `(` (different characters `?` can be replaced with different brackets). **You cannot reorder the characters, remove them, insert other characters, and each `?` must be replaced**.

Determine if it is possible to obtain an RBS after these replacements.

Input

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

Each test case consists of one line containing s $(2 \le |s| \le 100)$ — a sequence of characters `(`, `)`, and/or `?`. **There is exactly one character `(` and exactly one character `)`** in this sequence.

Output

For each test case, print YES if it is possible to obtain a regular bracket sequence, or NO otherwise}.

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

Example

input

```
5
()
(?)
(??)
??()
)?(?
```

output

```
YES
NO
YES
YES
NO
```

Note

In the first test case, the sequence is already an RBS.

In the third test case, you can obtain an RBS as follows: `()()` or `(())`.

In the fourth test case, you can obtain an RBS as follows: `()()`.

B. Red and Blue

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Monocarp had a sequence a consisting of $n + m$ integers $a_1, a_2, \dots, a_{n+m}$. He painted the elements into two colors, red and blue; n elements were painted red, all other m elements were painted blue.

After painting the elements, he has written two sequences $r_1, r_2, \dots, r_n$ and $b_1, b_2, \dots, b_m$. The sequence r consisted of all red elements of a **in the order they appeared in** a; similarly, the sequence b consisted of all blue elements of a **in the order they appeared in** a **as well**.

Unfortunately, the original sequence was lost, and Monocarp only has the sequences r and b. He wants to restore the original sequence. In case there are multiple ways to restore it, he wants to choose a way to restore that maximizes the value of

$$f(a) = \max(0, a_1, (a_1 + a_2), (a_1 + a_2 + a_3), \dots, (a_1 + a_2 + a_3 + \dots + a_{n+m}))$$

Help Monocarp to calculate the maximum possible value of $f(a)$.

Input

The first line contains one integer t $(1 \le t \le 1000)$ — the number of test cases. Then the test cases follow. Each test case consists of four lines.

The first line of each test case contains one integer n $(1 \le n \le 100)$.

The second line contains n integers $r_1, r_2, \dots, r_n$ $(-100 \le r_i \le 100)$.

The third line contains one integer m $(1 \le m \le 100)$.

The fourth line contains m integers $b_1, b_2, \dots, b_m$ $(-100 \le b_i \le 100)$.

Output

For each test case, print one integer — the maximum possible value of $f(a)$.

Example

input

```
4
4
6 -5 7 -3
3
2 3 -4
2
1 1
4
10 -3 2 2
5
-1 -2 -3 -4 -5
5
-1 -2 -3 -4 -5
1
0
1
0
```

output

```
13
13
0
0
```

Note

In the explanations for the sample test cases, red elements are marked as **bold**.

In the first test case, one of the possible sequences a is $[\mathbf{6}, 2, \mathbf{-5}, 3, \mathbf{7}, \mathbf{-3}, -4]$.

In the second test case, one of the possible sequences a is $[10, \mathbf{1}, -3, \mathbf{1}, 2, 2]$.

In the third test case, one of the possible sequences a is $[\mathbf{-1}, -1, -2, -3, \mathbf{-2}, -4, -5, \mathbf{-3}, \mathbf{-4}, \mathbf{-5}]$.

In the fourth test case, one of the possible sequences a is $[0, \mathbf{0}]$.

C. Building a Fence

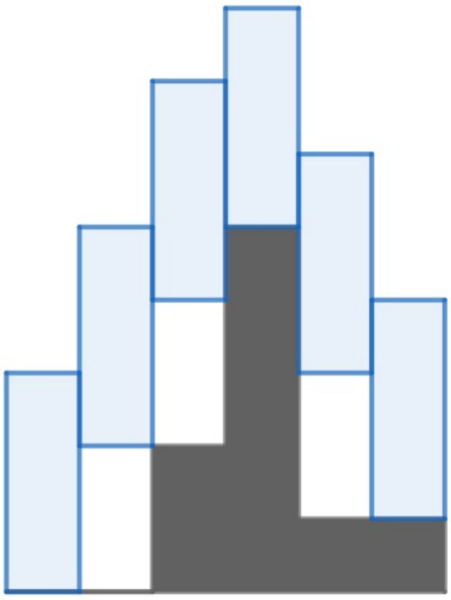
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You want to build a fence that will consist of n equal sections. All sections have a width equal to 1 and height equal to k. You will place all sections in one line side by side.

Unfortunately, the ground beneath the fence is not flat. For simplicity, you can think that the ground level under the i-th section is equal to h_i.

You should follow several rules to build the fence:

1. the consecutive sections should have a common side of length at least 1;
2. the first and the last sections should stand on the corresponding ground levels;
3. the sections between may be either on the ground level or higher, but not higher than $k-1$ from the ground level h_i (the height should be an integer);



One of possible fences (blue color) for the first test case

Is it possible to build a fence that meets all rules?

Input

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

The first line of each test case contains two integers n and k ($2 \le n \le 2 \cdot 10^5$; $2 \le k \le 10^8$) — the number of sections in the fence and the height of each section.

The second line of each test case contains n integers $h_1, h_2, \dots, h_n$ ($0 \le h_i \le 10^8$), where h_i is the ground level beneath the i-th section.

It's guaranteed that the sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output
For each test case print YES if it's possible to build the fence that meets all rules. Otherwise, print NO.

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

Example

input

```
3
6 3
0 0 2 5 1 1
2 3
0 2
3 2
3 0 2
```

output

```
YES
YES
NO
```

Note
In the first test case, one of the possible fences is shown in the picture.

In the second test case, according to the second rule, you should build both sections on the corresponding ground levels, and since $k = 3$, $h_1 = 0$, and $h_2 = 2$ the first rule is also fulfilled.

In the third test case, according to the second rule, you should build the first section on height 3 and the third section on height 2. According to the first rule, the second section should be on the height of at least 2 (to have a common side with the first section), but according to the third rule, the second section can be built on the height of at most $h_2 + k - 1 = 1$.

D. Ceil Divisions

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

You have an array $a_1, a_2, \dots, a_n$ where $a_i = i$.

In one step, you can choose two indices x and y ($x \neq y$) and set $a_x = \left\lceil \frac{a_x}{a_y} \right\rceil$ (ceiling function).

Your goal is to make array a consist of $n - 1$ ones and 1 two in no more than $n + 5$ steps. Note that you don't have to minimize the number of steps.

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

The first and only line of each test case contains the single integer n ($3 \le n \le 2 \cdot 10^5$) — the length of array a.

It's guaranteed that the sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output
For each test case, print the sequence of operations that will make a as $n - 1$ ones and 1 two in the following format: firstly, print one integer m ($m \le n + 5$) — the number of operations; next print m pairs of integers x and y ($1 \le x, y \le n$; $x \neq y$) (x may be greater or less than y) — the indices of the corresponding operation.

It can be proven that for the given constraints it's always possible to find a correct sequence of operations.

Example

input

```
2
3
4
```

output

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

Note
In the first test case, you have array $a = [1, 2, 3]$. For example, you can do the following:

1. choose 3, 2: $a_3 = \left\lceil \frac{a_3}{a_2} \right\rceil = 2$ and array $a = [1, 2, 2]$;
2. choose 3, 2: $a_3 = \left\lceil \frac{2}{2} \right\rceil = 1$ and array $a = [1, 2, 1]$.

You've got array with 2 ones and 1 two in 2 steps.
In the second test case, $a = [1, 2, 3, 4]$. For example, you can do the following:

1. choose 3, 4: $a_3 = \left\lceil \frac{3}{4} \right\rceil = 1$ and array $a = [1, 2, 1, 4]$;
2. choose 4, 2: $a_4 = \left\lceil \frac{4}{2} \right\rceil = 2$ and array $a = [1, 2, 1, 2]$;
3. choose 4, 2: $a_4 = \left\lceil \frac{2}{2} \right\rceil = 1$ and array $a = [1, 2, 1, 1]$.

E. A Bit Similar

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Let's call two strings a and b (both of length k) *a bit similar* if they have the same character in some position, i. e. there exists at least one $i \in [1, k]$ such that $a_i = b_i$.

You are given a binary string s of length n (a string of n characters `0` and/or `1`) and an integer k. Let's denote the string $s[i..j]$ as the substring of s starting from the i-th character and ending with the j-th character (that is, $s[i..j] = s_i s_{i+1} s_{i+2} \dots s_{j-1} s_j$).

Let's call a binary string t of length k *beautiful* if it is *a bit similar* to all substrings of s having length exactly k; that is, it is *a bit similar* to $s[1..k], s[2..k+1], \dots, s[n-k+1..n]$.

Your goal is to find the **lexicographically** smallest string t that is *beautiful*, or report that no such string exists. String x is lexicographically less than string y if either x is a prefix of y (and $x \neq y$), or there exists such i ($1 \le i \le \min(|x|, |y|)$), that $x_i < y_i$, and for any j ($1 \le j < i$) $x_j = y_j$.

Input

The first line contains one integer q ($1 \le q \le 10000$) — the number of test cases. Each test case consists of two lines.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 10^6$). The second line contains the string s, consisting of n characters (each character is either `0` or `1`).

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, print the answer as follows:

- if it is impossible to construct a *beautiful* string, print one line containing the string `NO` (note: **exactly in upper case**, you can't print `No`, for example);
- otherwise, print two lines. The first line should contain the string `YES` (exactly in upper case as well); the second line — the lexicographically smallest *beautiful* string, consisting of k characters `0` and/or `1`.

Example

input

```
7
4 2
0110
4 2
1001
9 3
010001110
9 3
101110001
10 3
0101110001
10 10
1111111111
11 10
11111111110
```

output

```
YES
11
YES
00
YES
010
YES
101
NO
YES
0000000001
YES
0000000010
```

F. Power Sockets

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

// We decided to drop the legend about the power sockets but feel free to come up with your own :^)

Define a chain:

- a chain of length 1 is a single vertex;
- a chain of length x is a chain of length $x - 1$ with a new vertex connected to the end of it with a single edge.

You are given n chains of lengths $l_1, l_2, \dots, l_n$. You plan to build a tree using some of them.

- Each vertex of the tree is either white or black.
- The tree initially only has a white root vertex.
- All chains initially consist only of white vertices.
- You can take one of the chains and connect any of its vertices to any white vertex of the tree with an edge. The chain becomes part of the tree. Both endpoints of this edge become black.
- Each chain can be used no more than once.
- Some chains can be left unused.

The distance between two vertices of the tree is the number of edges on the shortest path between them.

If there is at least k white vertices in the resulting tree, then the value of the tree is the distance between the root and the k-th closest white vertex.

What's the minimum value of the tree you can obtain? If there is no way to build a tree with at least k white vertices, then print -1.

Input

The first line contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $2 \le k \le 10^9$) — the number of chains and the minimum number of white vertices a tree should have to have a value.

The second line contains n integers $l_1, l_2, \dots, l_n$ ($3 \le l_i \le 2 \cdot 10^5$) — the lengths of the chains.

Output

Print a single integer. If there is no way to build a tree with at least k white vertices, then print -1. Otherwise, print the minimum value the tree can have.

Examples

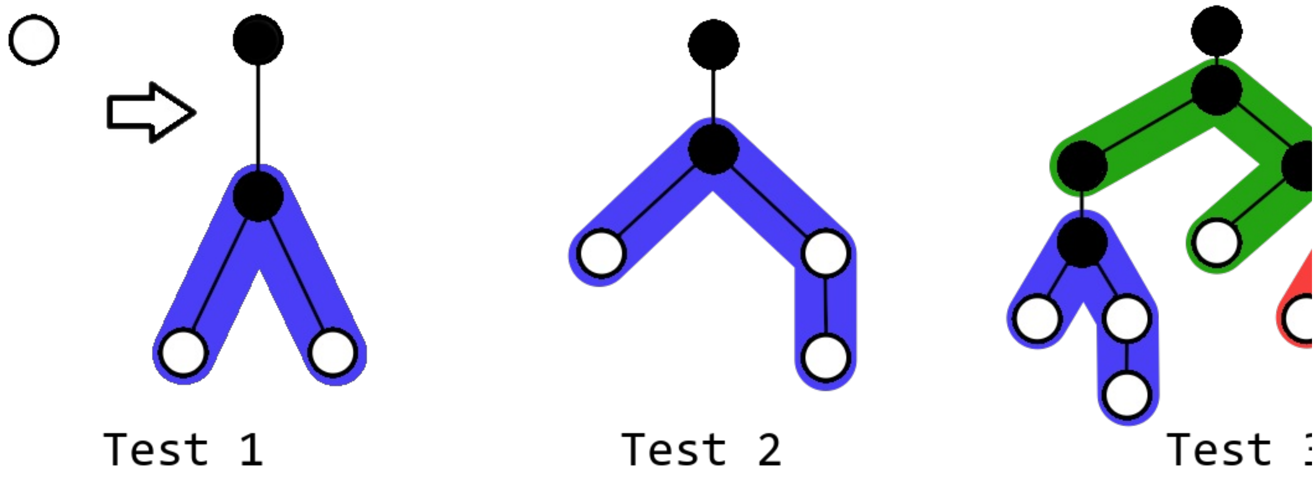
input
```
1 2
3
```
output
```
2
```

input
```
3 3
4 3 3
```
output
```
3
```

input
```
3 5
4 3 4
```
output
```
4
```

input
```
2 10
5 7
```
output
```
-1
```

Note



You are allowed to not use all the chains, so it's optimal to only use chain of length 4 in the second example.