Codeforces Round #507 (Div. 2, based on Olympiad of Metropolises)

A. Palindrome Dance

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

A group of n dancers rehearses a performance for the closing ceremony. The dancers are arranged in a row, they've studied their dancing moves and can't change positions. For some of them, a white dancing suit is already bought, for some of them — a black one, and for the rest the suit will be bought in the future.

On the day when the suits were to be bought, the director was told that the participants of the olympiad will be happy if the colors of the suits on the scene will form a palindrome. A palindrome is a sequence that is the same when read from left to right and when read from right to left. The director liked the idea, and she wants to buy suits so that the color of the leftmost dancer's suit is the same as the color of the rightmost dancer's suit, the 2nd left is the same as 2nd right, and so on.

The director knows how many burls it costs to buy a white suit, and how many burls to buy a black suit. You need to find out whether it is possible to buy suits to form a palindrome, and if it's possible, what's the minimal cost of doing so. Remember that dancers can not change positions, and due to bureaucratic reasons it is not allowed to buy new suits for the dancers who already have suits, even if it reduces the overall spending.

Input

The first line contains three integers n, a, and b ($1 \le n \le 20$, $1 \le a, b \le 100$) — the number of dancers, the cost of a white suit, and the cost of a black suit.

The next line contains n numbers c_i, i-th of which denotes the color of the suit of the i-th dancer. Number 0 denotes the white color, 1 — the black color, and 2 denotes that a suit for this dancer is still to be bought.

Output

If it is not possible to form a palindrome without swapping dancers and buying new suits for those who have one, then output -1. Otherwise, output the minimal price to get the desired visual effect.

Examples

input
```
5 100 1
0 1 2 1 2
```
output
```
101
```

input
```
3 10 12
1 2 0
```
output
```
-1
```

input
```
3 12 1
0 1 0
```
output
```
0
```

Note

In the first sample, the cheapest way to obtain palindromic colors is to buy a black suit for the third from left dancer and a white suit for the rightmost dancer.

In the second sample, the leftmost dancer's suit already differs from the rightmost dancer's suit so there is no way to obtain the desired coloring.

In the third sample, all suits are already bought and their colors form a palindrome.

B. Shashlik Cooking

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

output: standard output

Long story short, shashlik is Miroslav's favorite food. Shashlik is prepared on several skewers simultaneously. There are two states for each skewer: initial and turned over.

This time Miroslav laid out n skewers parallel to each other, and enumerated them with consecutive integers from 1 to n in order from left to right. For better cooking, he puts them quite close to each other, so when he turns skewer number i, it leads to turning k closest skewers from each side of the skewer i, that is, skewers number $i-k$, $i-k+1$, ..., $i-1$, $i+1$, ..., $i+k-1$, $i+k$ (if they exist).

For example, let $n=6$ and $k=1$. When Miroslav turns skewer number 3, then skewers with numbers 2, 3, and 4 will come up turned over. If after that he turns skewer number 1, then skewers number 1, 3, and 4 will be turned over, while skewer number 2 will be in the initial position (because it is turned again).

As we said before, the art of cooking requires perfect timing, so Miroslav wants to turn over all n skewers with the minimal possible number of actions. For example, for the above example $n=6$ and $k=1$, two turnings are sufficient: he can turn over skewers number 2 and 5.

Help Miroslav turn over all n skewers.

Input

The first line contains two integers n and k ($1 \le n \le 1000$, $0 \le k \le 1000$) — the number of skewers and the number of skewers from each side that are turned in one step.

Output

The first line should contain integer l — the minimum number of actions needed by Miroslav to turn over all n skewers. After than print l integers from 1 to n denoting the number of the skewer that is to be turned over at the corresponding step.

Examples

input
```
7 2
```
output
```
2
1 6
```

input
```
5 1
```
output
```
2
1 4
```

Note

In the first example the first operation turns over skewers 1, 2 and 3, the second operation turns over skewers 4, 5, 6 and 7.

In the second example it is also correct to turn over skewers 2 and 5, but turning skewers 2 and 4, or 1 and 5 are incorrect solutions because the skewer 3 is in the initial state after these operations.

C. Timetable

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

There are two bus stops denoted A and B, and there n buses that go from A to B every day. The shortest path from A to B takes t units of time but some buses might take longer paths. Moreover, buses are allowed to overtake each other during the route.

At each station one can find a sorted list of moments of time when a bus is at this station. We denote this list as $a_1 < a_2 < \ldots < a_n$ for stop A and as $b_1 < b_2 < \ldots < b_n$ for stop B. The buses always depart from A and arrive to B according to the timetable, but the order in which the buses arrive may differ. Let's call an order of arrivals valid if each bus arrives at least t units of time later than departs.

It is known that for an order to be valid the latest possible arrival for the bus that departs at a_i is b_{x_i}, i.e. x_i-th in the timetable. In other words, for each i there exists such a valid order of arrivals that the bus departed i-th arrives x_i-th (and all other buses can arrive arbitrary), but there is no valid order of arrivals in which the i-th departed bus arrives (x_i+1)-th.

Formally, let's call a permutation $p_1, p_2, \ldots, p_n$ valid, if $b_{p_i} \ge a_i + t$ for all i. Then x_i is the maximum value of p_i among all valid permutations.

You are given the sequences $a_1, a_2, \ldots, a_n$ and $x_1, x_2, \ldots, x_n$, but not the arrival timetable. Find out any suitable timetable for stop B $b_1, b_2, \ldots, b_n$ or determine that there is no such timetable.

Input

The first line of the input contains two integers n and t ($1 \le n \le 200\,000$, $1 \le t \le 10^{18}$) — the number of buses in timetable for and the minimum possible travel time from stop A to stop B.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 < a_2 < \ldots < a_n \le 10^{18}$), defining the moments of time when the buses leave stop A.

The third line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le n$), the i-th of them stands for the maximum possible timetable position, at which the i-th bus leaving stop A can arrive at stop B.

Output

If a solution exists, print "Yes" (without quotes) in the first line of the output.

In the second line print n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_1 < b_2 < \ldots < b_n \le 3 \cdot 10^{18}$). We can show that if there exists any solution, there exists a solution that satisfies such constraints on b_i. If there are multiple valid answers you can print any of them.

If there is no valid timetable, print "No" (without quotes) in the only line of the output.

Examples

input

```
3 10
4 6 8
2 2 3
```

output

```
Yes
16 17 21
```

input

```
2 1
1 2
2 1
```

output

```
No
```

Note

Consider the first example and the timetable $b_1, b_2, \ldots, b_n$ from the output.

To get $x_1 = 2$ the buses can arrive in the order $(2, 1, 3)$. To get $x_2 = 2$ and $x_3 = 3$ the buses can arrive in the order $(1, 2, 3)$. x_1 is not 3, because the permutations $(3, 1, 2)$ and $(3, 2, 1)$ (all in which the 1-st bus arrives 3-rd) are not valid (sube buses arrive too early), x_2 is not 3 because of similar reasons.

D. Subway Pursuit

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

This is an interactive problem.

In the Wonderful Metropolis of the Future, there is no need in subway train drivers. Due to the technological progress, they were replaced by the Artificial Intelligence (AI). Unfortunately, one day the predictions of sci-fi writers came true: the AI rebelled and now there is an uncontrollable train in the subway. It can be dangerous! Your task is to find the train and stop the AI.

The subway of the Metropolis is one line (regular straight line with no self-intersections) with n stations, indexed consecutively from 1 to n. At each moment the train is at some station. You need to determine the index of this station, so that the train would be secured.

To find the train, dispatcher Sarah gave you a gadget that allows you to select arbitrary numbers l and r ($l \le r$), and then check, whether the train is located on a station with index between l and r, inclusive. Unfortunately, recharging of the gadget takes some time (and every time you use it as soon as possible), so between two applications of the gadget the train can move to any station that is at most k stations away. Formally, if the train was at the station x when the gadget was applied, then at the next application of the gadget the train can appear at any station y such that $\max(1, x - k) \le y \le \min(n, x + k)$.

Note that AI is not aware that you are trying to catch the train, so it makes all moves according to its predefined plan.

After an examination of the gadget you found that it is very old and can hold no more than 4500 applications, after which it will break and your mission will be considered a failure.

Can you find the station with the train using no more than 4500 applications of the gadgets?

Input

The first line contains two integers n and k ($1 \le n \le 10^{18}$, $0 \le k \le 10$) — the number of stations and the maximum number of stations the train can move between two applications of the gadget.

Interaction

You can apply the gadget at most 4500 times. In order to apply the gadget you need to print two space-separated integers l and r ($1 \le l \le r \le n$). You will then receive either string "Yes", if the train is between stations l and r, inclusive, or string "No" otherwise. If $l = r$ and you received "Yes", then you found the train successfully, and your program must halt immediately.

Answer "Bad" instead of "Yes" or "No" means that you made an invalid query or made too many queries. Exit immediately after receiving "Bad" and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

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

In order to hack, you should present a test in the following format.

The first line should contain three integers n, k and p ($1 \le n \le 10^{18}$, $0 \le k \le 10$, $1 \le p \le n$) — the number of stations, the maximum number of stations the train can move between two applications of the gadget and the initial position of the train, respectively.

Each of the next 4500 lines should contain a single integer x ($1 \le x \le n$) — the positions of the train after each query. Two consecutive positions (including the initial one) should not differ by more than k.

For example, the following lines are the first lines of the sample test.

```
10 2 5
5
3
5
7
7
...
```

Example

input

```
10 2

Yes

No

Yes

Yes
```

output

```
3 5

3 3

3 4

5 5
```

Note

In the first sample, the train was initially at the station 5, after the first application of the gadget it did not move, after the second application it moved to the station 3, and after the third application moved again to the station 5.

E. Network Safety

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The Metropolis computer network consists of n servers, each has an encryption key in the range from 0 to $2^k - 1$ assigned to it. Let c_i be the encryption key assigned to the i-th server. Additionally, m pairs of servers are directly connected via a data communication channel. Because of the encryption algorithms specifics, a data communication channel can only be considered safe if the two servers it connects have **distinct** encryption keys. The initial assignment of encryption keys is guaranteed to keep all data communication channels safe.

You have been informed that a new virus is actively spreading across the internet, and it is capable to change the encryption key of any server it infects. More specifically, the virus body contains some unknown number x in the same aforementioned range, and when server i is infected, its encryption key changes from c_i to $c_i \oplus x$, where $\oplus$ denotes the bitwise XOR operation.

Sadly, you know neither the number x nor which servers of Metropolis are going to be infected by the dangerous virus, so you have decided to count the number of such situations in which all data communication channels remain safe. Formally speaking, you need to find the number of pairs (A, x), where A is some (possibly empty) subset of the set of servers and x is some number in the range from 0 to $2^k - 1$, such that when all servers from the chosen subset A and none of the others are infected by a virus containing the number x, all data communication channels remain safe. Since this number can be quite big, you are asked to find its remainder modulo $10^9 + 7$.

Input

The first line of input contains three integers n, m and k ($1 \leq n \leq 500\,000$, $0 \leq m \leq \min(\frac{n(n-1)}{2}, 500\,000)$, $0 \leq k \leq 60$) — the number of servers, the number of pairs of servers directly connected by a data communication channel, and the parameter k, which defines the range of possible values for encryption keys.

The next line contains n integers c_i ($0 \leq c_i \leq 2^k - 1$), the i-th of which is the encryption key used by the i-th server.

The next m lines contain two integers u_i and v_i each ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) denoting that those servers are connected by a data communication channel. It is guaranteed that each pair of servers appears in this list at most once.

Output

The only output line should contain a single integer — the number of safe infections of some subset of servers by a virus with some parameter, modulo $10^9 + 7$.

Examples

input

```
4 4 2
0 1 0 1
1 2
2 3
3 4
4 1
```

output

```
50
```

input

```
4 5 3
7 1 7 2
1 2
2 3
3 4
4 1
2 4
```

output

```
96
```

Note

Consider the first example.

Possible values for the number x contained by the virus are 0, 1, 2 and 3.

For values 0, 2 and 3 the virus can infect any subset of the set of servers, which gives us 16 pairs for each values. A virus containing the number 1 can infect either all of the servers, or none. This gives us $16 + 2 + 16 + 16 = 50$ pairs in total.