Codeforces Round #176 (Div. 1)

A. Lucky Permutation

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

A permutation p of size n is the sequence $p_1, p_2, ..., p_n$, consisting of n distinct integers, each of them is from 1 to n $(1 \leq p_i \leq n)$.

A lucky permutation is such permutation p, that any integer i $(1 \leq i \leq n)$ meets this condition $p_{p_i} = n - i + 1$.

You have integer n. Find some lucky permutation p of size n.

Input

The first line contains integer n $(1 \leq n \leq 10^5)$ — the required permutation size.

Output

Print "-1" (without the quotes) if the lucky permutation p of size n doesn't exist.

Otherwise, print n distinct integers $p_1, p_2, ..., p_n$ $(1 \leq p_i \leq n)$ after a space — the required permutation.

If there are multiple answers, you can print any of them.

Sample test(s)

input
1
output
1

input
2
output
-1

input
4
output
2 4 1 3

input
5
output
2 5 3 1 4

B. Shifting

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

John Doe has found the beautiful permutation formula.

Let's take permutation $p = p_1, p_2, ..., p_n$. Let's define transformation f of this permutation:

$$f(p, k) = \overbrace{p_2, p_3, \ldots, p_k, p_1}, \overbrace{p_{k+2}, p_{k+3}, \ldots, p_{2k}, p_{k+1}}, \ldots, \overbrace{p_{rk+2}, p_{rk+3}, \ldots, p_n, p_{rk+1}};$$

where $k\ (k > 1)$ is an integer, the transformation parameter, r is such maximum integer that $rk \le n$. If $rk = n$, then elements p_{rk+1}, p_{rk+2} and so on are omitted. In other words, the described transformation of permutation p cyclically shifts to the left each consecutive block of length k and the last block with the length equal to the remainder after dividing n by k.

John Doe thinks that permutation $f(f(\ldots f(p = [1, 2, ..., n], 2) \ldots, n - 1), n)$ is beautiful. Unfortunately, he cannot quickly find the beautiful permutation he's interested in. That's why he asked you to help him.

Your task is to find a beautiful permutation for the given n. For clarifications, see the notes to the third sample.

Input

A single line contains integer n $(2 \le n \le 10^6)$.

Output

Print n distinct space-separated integers from 1 to n — a beautiful permutation of size n.

Sample test(s)

input
2
output
2 1

input
3
output
1 3 2

input
4
output
4 2 3 1

Note

A note to the third test sample:

- $f([1, 2, 3, 4], 2) = [2, 1, 4, 3]$
- $f([2, 1, 4, 3], 3) = [1, 4, 2, 3]$
- $f([1, 4, 2, 3], 4) = [4, 2, 3, 1]$

C. Main Sequence

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

As you know, Vova has recently become a new shaman in the city of Ultima Thule. So, he has received the shaman knowledge about the correct bracket sequences. The shamans of Ultima Thule have been using lots of different types of brackets since prehistoric times. A bracket type is a positive integer. The shamans define a correct bracket sequence as follows:

- An empty sequence is a correct bracket sequence.
- If $\{a_1, a_2, ..., a_l\}$ and $\{b_1, b_2, ..., b_k\}$ are correct bracket sequences, then sequence $\{a_1, a_2, ..., a_l, b_1, b_2, ..., b_k\}$ (their concatenation) also is a correct bracket sequence.
- If $\{a_1, a_2, ..., a_l\}$ — is a correct bracket sequence, then sequence $\{v, a_1, a_2, \ldots, a_l, -v\}$ also is a correct bracket sequence, where v $(v > 0)$ is an integer.

For example, sequences $\{1, 1, -1, 2, -2, -1\}$ and $\{3, -3\}$ are correct bracket sequences, and $\{2, -3\}$ is not.

Moreover, after Vova became a shaman, he learned the *most important* correct bracket sequence $\{x_1, x_2, ..., x_n\}$, consisting of n integers. As sequence x is the most important, Vova decided to encrypt it just in case.

Encrypting consists of two sequences. The first sequence $\{p_1, p_2, ..., p_n\}$ contains types of brackets, that is, $p_i = |x_i|$ $(1 \le i \le n)$. The second sequence $\{q_1, q_2, ..., q_t\}$ contains t integers — **some positions** (possibly, not all of them), which had negative numbers in sequence $\{x_1, x_2, ..., x_n\}$.

Unfortunately, Vova forgot the main sequence. But he was lucky enough to keep the encryption: sequences $\{p_1, p_2, ..., p_n\}$ and $\{q_1, q_2, ..., q_t\}$. Help Vova restore sequence x by the encryption. If there are multiple sequences that correspond to the encryption, restore any of them. If there are no such sequences, you should tell so.

Input

The first line of the input contains integer n $(1 \le n \le 10^6)$. The second line contains n integers: $p_1, p_2, ..., p_n$ $(1 \le p_i \le 10^9)$.

The third line contains integer t $(0 \le t \le n)$, followed by t distinct integers $q_1, q_2, ..., q_t$ $(1 \le q_i \le n)$.

The numbers in each line are separated by spaces.

Output

Print a single string "NO" (without the quotes) if Vova is mistaken and a suitable sequence $\{x_1, x_2, ..., x_n\}$ doesn't exist.

Otherwise, in the first line print "YES" (without the quotes) and in the second line print n integers $x_1, x_2, ..., x_n$ $(|x_i| = p_i; x_{q_j} < 0)$. If there are multiple sequences that correspond to the encrypting, you are allowed to print any of them.

Sample test(s)

input
2 1 1 0
output
YES 1 -1

input
4 1 1 1 1 1 3
output
YES 1 1 -1 -1

input
3 1 1 1 0
output
NO

input
4 1 2 2 1 2 3 4

output

```
YES
1 2 -2 -1
```

D. Tourists

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

A double tourist path, located at a park in Ultima Thule, is working by the following principle:

- We introduce the Cartesian coordinate system.
- At some points of time there are two tourists going (for a walk) from points $(-1, 0)$ and $(1, 0)$ simultaneously. The first one is walking from $(-1, 0)$, the second one is walking from $(1, 0)$.
- Both tourists in a pair move at the same speed 1 (distance unit per second), the first one moves along line $x = -1$, the second one moves along line $x = 1$, both of them are moving in the positive direction of the Oy axis.
- At some points of time walls appear. Wall (l_i, r_i) is a segment between points $(0, l_i)$ and $(0, r_i)$. Each wall appears immediately.

The Ultima Thule government wants to learn this for each pair of tourists that walk simultaneously: for how long (in seconds) will they not see each other? Two tourists don't see each other if the segment that connects their positions on the plane intersects at least one wall. Two segments intersect if they share at least one point. We assume that the segments' ends belong to the segments.

Help the government count the required time. Note that the walls can intersect (in any way) or coincide.

Input

The first line contains two space-separated integers n and m ($1 \le n, m \le 10^5$) — the number of pairs of tourists and the number of built walls. The next m lines contain three space-separated integers l_i, r_i and t_i each ($0 \le l_i < r_i \le 10^9$, $0 \le t_i \le 10^9$) — the wall ends and the time it appeared. The last line contains n distinct space-separated strictly increasing integers $q_1, q_2, ..., q_n$ ($0 \le q_i \le 10^9$) — the points of time when pairs of tourists walk.

All points of time are given in seconds.

Output

For each pair of tourists print on a single line a single integer — the time in seconds when the two tourists from the corresponding pair won't see each other. Print the numbers in the order in which the they go in the input.

Sample test(s)

input

```
2 2
1 4 3
3 6 5
0 1
```

output

```
2
4
```

input

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

output

```
2
4
4
```

E. Ladies' Shop

time limit per test: 8 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A ladies' shop has recently opened in the city of Ultima Thule. To get ready for the opening, the shop bought n bags. Each bag is characterised by the total weight a_i of the items you can put there. The weird thing is, you **cannot** use these bags to put a set of items with the total weight strictly less than a_i. However the weights of the items that will be sold in the shop haven't yet been defined. That's what you should determine right now.

Your task is to find the set of the items' weights $p_1, p_2, ..., p_k$ $(1 \le p_1 < p_2 < ... < p_k)$, such that:

1. Any bag will be used. That is, for any i $(1 \le i \le n)$ there will be such set of items that their total weight will equal a_i. We assume that there is the infinite number of items of any weight. You can put multiple items of the same weight in one bag.
2. For any set of items that have total weight less than or equal to m, there is a bag into which you can put this set. Similarly, a set of items can contain multiple items of the same weight.
3. Of all sets of the items' weights that satisfy points 1 and 2, find the set with the minimum number of weights. In other words, value k should be as small as possible.

Find and print the required set.

Input

The first line contains space-separated integers n and m $(1 \le n, m \le 10^6)$. The second line contains n distinct space-separated integers $a_1, a_2, ..., a_n$ $(1 \le a_1 < a_2 < ... < a_n \le m)$ — the bags' weight limits.

Output

In the first line print "NO" (without the quotes) if there isn't set p_i, that would meet the conditions.

Otherwise, in the first line print "YES" (without the quotes), in the second line print an integer k (showing how many numbers are in the suitable set with the minimum number of weights), in the third line print k space-separated integers $p_1, p_2, ..., p_k$ $(1 \le p_1 < p_2 < ... < p_k)$. If there are multiple solutions, print any of them.

Sample test(s)

input
6 10 5 6 7 8 9 10

output
YES 5 5 6 7 8 9

input
1 10 1

output
NO

input
1 10 6

output
YES 1 6