A. Repeating Cipher

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

Polycarp loves ciphers. He has invented his own cipher called *repeating*.

Repeating cipher is used for strings. To encrypt the string $s = s_1 s_2 \dots s_m$ $(1 \le m \le 10)$, Polycarp uses the following algorithm:

- he writes down s_1 ones,
- he writes down s_2 twice,
- he writes down s_3 three times,
- ...
- he writes down s_m m times.

For example, if s="bab" the process is: "b" $\to$ "baa" $\to$ "baabbb". So the encrypted s="bab" is "baabbb".

Given string t — the result of encryption of some string s. Your task is to decrypt it, i. e. find the string s.

Input

The first line contains integer n $(1 \le n \le 55)$ — the length of the encrypted string. The second line of the input contains t — the result of encryption of some string s. It contains only lowercase Latin letters. The length of t is exactly n.

It is guaranteed that the answer to the test exists.

Output

Print such string s that after encryption it equals t.

Examples

input
```
6
baabbb
```
output
```
bab
```

input
```
10
ooopppssss
```
output
```
oops
```

input
```
1
z
```
output
```
z
```

B. Array Stabilization

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

You are given an array a consisting of n integer numbers.

Let *instability* of the array be the following value: $\max\limits_{i=1}^{n} a_i - \min\limits_{i=1}^{n} a_i$.

You have to remove **exactly one** element from this array to minimize *instability* of the resulting $(n-1)$-elements array. Your task is to calculate the minimum possible *instability*.

Input

The first line of the input contains one integer n ($2 \le n \le 10^5$) — the number of elements in the array a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^5$) — elements of the array a.

Output

Print one integer — the minimum possible *instability* of the array if you have to remove **exactly one** element from the array a.

Examples

input
4 1 3 3 7
output
2

input
2 1 100000
output
0

Note

In the first example you can remove 7 then *instability* of the remaining array will be $3 - 1 = 2$.

In the second example you can remove either 1 or 100000 then *instability* of the remaining array will be $100000 - 100000 = 0$ and $1 - 1 = 0$ correspondingly.

C. Powers Of Two

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

A positive integer x is called a *power of two* if it can be represented as $x = 2^y$, where y is a non-negative integer. So, the *powers of two* are $1, 2, 4, 8, 16, \dots$.

You are given two positive integers n and k. Your task is to represent n as the **sum** of **exactly** k *powers of two*.

Input

The only line of the input contains two integers n and k ($1 \le n \le 10^9$, $1 \le k \le 2 \cdot 10^5$).

Output

If it is impossible to represent n as the sum of k powers of two, print `NO`.

Otherwise, print YES, and then print k positive integers $b_1, b_2, \dots, b_k$ such that each of b_i is a power of two, and $\sum\limits_{i=1}^{k} b_i = n$. If there are multiple answers, you may print any of them.

Examples

input
9 4
output
YES 1 2 2 4

input
8 1
output
YES 8

input
5 1
output
NO

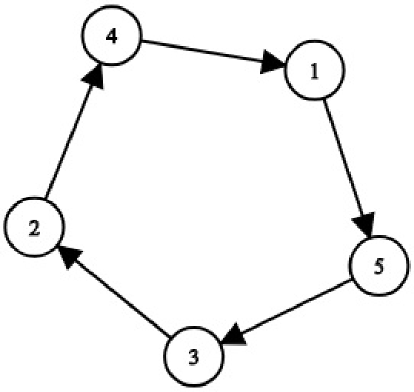
input
3 7

output
NO

D. Circular Dance

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n kids, numbered from 1 to n, dancing in a circle around the Christmas tree. Let's enumerate them in a clockwise direction as p_1, p_2, ..., p_n (all these numbers are from 1 to n and are distinct, so p is a permutation). Let the next kid for a kid p_i be kid p_{i+1} if $i < n$ and p_1 otherwise. After the dance, each kid remembered two kids: the next kid (let's call him x) and the next kid for x. Each kid told you which kids he/she remembered: the kid i remembered kids $a_{i,1}$ and $a_{i,2}$. However, the order of $a_{i,1}$ and $a_{i,2}$ can differ from their order in the circle.



Example: 5 kids in a circle, $p = [3, 2, 4, 1, 5]$ (or any cyclic shift). The information kids remembered is: $a_{1,1} = 3$, $a_{1,2} = 5$; $a_{2,1} = 1$, $a_{2,2} = 4$; $a_{3,1} = 2$, $a_{3,2} = 4$; $a_{4,1} = 1$, $a_{4,2} = 5$; $a_{5,1} = 2$, $a_{5,2} = 3$.

You have to restore the order of the kids in the circle using this information. If there are several answers, you may print any. It is guaranteed that at least one solution exists.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input

The first line of the input contains one integer n ($3 \le n \le 2 \cdot 10^5$) — the number of the kids.

The next n lines contain 2 integers each. The i-th line contains two integers $a_{i,1}$ and $a_{i,2}$ ($1 \le a_{i,1}, a_{i,2} \le n, a_{i,1} \ne a_{i,2}$) — the kids the i-th kid remembered, given in arbitrary order.

Output

Print n integers p_1, p_2, ..., p_n — permutation of integers from 1 to n, which corresponds to the order of kids in the circle. **If there are several answers, you may print any** (for example, it doesn't matter which kid is the first in the circle). It is guaranteed that at least one solution exists.

Examples

input
5 3 5 1 4 2 4 1 5 2 3
output
3 2 4 1 5

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

E. Almost Regular Bracket Sequence

time limit per test: 3 seconds

memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a bracket sequence s consisting of n opening '(' and closing ')' brackets.

A *regular* bracket sequence is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters '1' and '+' between the original characters of the sequence. For example, bracket sequences "()()", "(())" are regular (the resulting expressions are: "(1)+(1)", "((1+1)+1)"), and ")(" and "(" are not.

You can change the type of some bracket s_i. It means that if $s_i =$ ')' then you can change it to '(' and vice versa.

Your task is to calculate the number of positions i such that if you change the type of the i-th bracket, then the resulting bracket sequence becomes *regular*.

Input

The first line of the input contains one integer n ($1 \le n \le 10^6$) — the length of the bracket sequence.

The second line of the input contains the string s consisting of n opening '(' and closing ')' brackets.

Output

Print one integer — the number of positions i such that if you change the type of the i-th bracket, then the resulting bracket sequence becomes *regular*.

Examples

input

```
6
(((())
```

output

```
3
```

input

```
6
()()()
```

output

```
0
```

input

```
1
)
```

output

```
0
```

input

```
8
)))(((((
```

output

```
0
```

F. Make It Connected

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

You are given an undirected graph consisting of n vertices. A number is written on each vertex; the number on vertex i is a_i. Initially there are no edges in the graph.

You may add some edges to this graph, but you have to pay for them. The cost of adding an edge between vertices x and y is $a_x + a_y$ coins. There are also m special offers, each of them is denoted by three numbers x, y and w, and means that you can add an edge connecting vertices x and y and pay w coins for it. You don't have to use special offers: if there is a pair of vertices x and y that has a special offer associated with it, you still may connect these two vertices paying $a_x + a_y$ coins for it.

What is the minimum number of coins you have to spend to make the graph connected? Recall that a graph is connected if it's possible to get from any vertex to any other vertex using only the edges belonging to this graph.

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the number of vertices in the graph and the number of special offers, respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^{12}$) — the numbers written on the vertices.

Then m lines follow, each containing three integers x, y and w ($1 \le x, y \le n$, $1 \le w \le 10^{12}$, $x \ne y$) denoting a special offer: you may add an edge connecting vertex x and vertex y, and this edge will cost w coins.

Output

Print one integer — the minimum number of coins you have to pay to make the graph connected.

Examples

input

```
3 2
1 3 3
2 3 5
2 1 1
```

output

```
5
```

input

```
4 0
1 3 3 7
```

output

```
16
```

input

```
5 4
1 2 3 4 5
1 2 8
1 3 10
1 4 7
1 5 15
```

output

```
18
```

Note

In the first example it is possible to connect 1 to 2 using special offer 2, and then 1 to 3 without using any offers.

In next two examples the optimal answer may be achieved without using special offers.