Avito Cool Challenge 2018

A. Definite Game

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

Chouti was doing a competitive programming competition. However, after having all the problems accepted, he got bored and decided to invent some small games.

He came up with the following game. The player has a positive integer n. Initially the value of n equals to v and the player is able to do the following operation as many times as the player want (possibly zero): choose a positive integer x that $x < n$ and x is not a divisor of n, then subtract x from n. The goal of the player is to minimize the value of n in the end.

Soon, Chouti found the game trivial. Can you also beat the game?

Input
The input contains only one integer in the first line: v ($1 \leq v \leq 10^9$), the initial value of n.

Output
Output a single integer, the minimum value of n the player can get.

Examples

input
8
output
1

input
1
output
1

Note
In the first example, the player can choose $x = 3$ in the first turn, then n becomes 5. He can then choose $x = 4$ in the second turn to get $n = 1$ as the result. There are other ways to get this minimum. However, for example, he cannot choose $x = 2$ in the first turn because 2 is a divisor of 8.

In the second example, since $n = 1$ initially, the player can do nothing.

B. Farewell Party

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

Chouti and his classmates are going to the university soon. To say goodbye to each other, the class has planned a big farewell party in which classmates, teachers and parents sang and danced.

Chouti remembered that n persons took part in that party. To make the party funnier, each person wore one hat among n kinds of weird hats numbered $1, 2, \ldots n$. It is possible that several persons wore hats of the same kind. Some kinds of hats can remain unclaimed by anyone.

After the party, the i-th person said that there were a_i persons wearing a hat **differing from his own**.

It has been some days, so Chouti forgot all about others' hats, but he is curious about that. Let b_i be the number of hat type the i-th person was wearing, Chouti wants you to find any possible $b_1, b_2, \ldots, b_n$ that doesn't contradict with any person's statement. Because some persons might have a poor memory, there could be no solution at all.

Input
The first line contains a single integer n ($1 \leq n \leq 10^5$), the number of persons in the party.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq n - 1$), the statements of people.

Output

If there is no solution, print a single line "Impossible".

Otherwise, print "Possible" and then n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$).

If there are multiple answers, print any of them.

Examples

input
3 0 0 0
output
Possible 1 1 1

input
5 3 3 2 2 2
output
Possible 1 1 2 2 2

input
4 0 1 2 3
output
Impossible

Note

In the answer to the first example, all hats are the same, so every person will say that there were no persons wearing a hat different from kind 1.

In the answer to the second example, the first and the second person wore the hat with type 1 and all other wore a hat of type 2.

So the first two persons will say there were three persons with hats differing from their own. Similarly, three last persons will say there were two persons wearing a hat different from their own.

In the third example, it can be shown that no solution exists.

In the first and the second example, other possible configurations are possible.

C. Colorful Bricks

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

On his free time, Chouti likes doing some housework. He has got one new task, paint some bricks in the yard.

There are n bricks lined in a row on the ground. Chouti has got m paint buckets of different colors at hand, so he painted each brick in one of those m colors.

Having finished painting all bricks, Chouti was satisfied. He stood back and decided to find something fun with these bricks. After some counting, he found there are k bricks with a color different from the color of the brick on its left (the first brick is not counted, for sure).

So as usual, he needs your help in counting how many ways could he paint the bricks. Two ways of painting bricks are different if there is at least one brick painted in different colors in these two ways. Because the answer might be quite big, you only need to output the number of ways modulo $998\,244\,353$.

Input

The first and only line contains three integers n, m and k ($1 \le n, m \le 2000, 0 \le k \le n-1$) — the number of bricks, the number of colors, and the number of bricks, such that its color differs from the color of brick to the left of it.

Output

Print one integer — the number of ways to color bricks modulo $998\,244\,353$.

Examples

input
3 3 0
output
3

input
3 2 1
output
4

Note

In the first example, since $k = 0$, the color of every brick should be the same, so there will be exactly $m = 3$ ways to color the bricks.

In the second example, suppose the two colors in the buckets are yellow and lime, the following image shows all 4 possible colorings.

D. Maximum Distance

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

Chouti was tired of the tedious homework, so he opened up an old programming problem he created years ago.

You are given a connected undirected graph with n vertices and m weighted edges. There are k special vertices: $x_1, x_2, \ldots, x_k$.

Let's define the cost of the path as the **maximum** weight of the edges in it. And the *distance* between two vertexes as the **minimum** cost of the paths connecting them.

For each special vertex, find another special vertex which is farthest from it (in terms of the previous paragraph, i.e. the corresponding *distance* is maximum possible) and output the distance between them.

The original constraints are really small so he thought the problem was boring. Now, he raises the constraints and hopes you can solve it for him.

Input

The first line contains three integers n, m and k ($2 \leq k \leq n \leq 10^5$, $n - 1 \leq m \leq 10^5$) — the number of vertices, the number of edges and the number of special vertices.

The second line contains k distinct integers $x_1, x_2, \ldots, x_k$ ($1 \leq x_i \leq n$).

Each of the following m lines contains three integers u, v and w ($1 \leq u, v \leq n, 1 \leq w \leq 10^9$), denoting there is an edge between u and v of weight w. The given graph is undirected, so an edge (u, v) can be used in the both directions.

The graph may have multiple edges and self-loops.

It is guaranteed, that the graph is connected.

Output

The first and only line should contain k integers. The i-th integer is the distance between x_i and the farthest special vertex from it.

Examples

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

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

Note

In the first example, the distance between vertex 1 and 2 equals to 2 because one can walk through the edge of weight 2

connecting them. So the distance to the farthest node for both 1 and 2 equals to 2.

In the second example, one can find that distance between 1 and 2, distance between 1 and 3 are both 3 and the distance between 2 and 3 is 2.

The graph may have multiple edges between and self-loops, as in the first example.

E. Missing Numbers

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

Chouti is working on a strange math problem.

There was a sequence of n positive integers $x_1, x_2, \ldots, x_n$, where n is even. The sequence was very special, namely for every integer t from 1 to n, $x_1 + x_2 + \ldots + x_t$ is a square of some integer number (that is, a perfect square).

Somehow, the numbers with odd indexes turned to be missing, so he is only aware of numbers on even positions, i.e. $x_2, x_4, x_6, \ldots, x_n$. The task for him is to restore the original sequence. Again, it's your turn to help him.

The problem setter might make mistakes, so there can be no possible sequence at all. If there are several possible sequences, you can output any.

Input

The first line contains an even number n ($2 \le n \le 10^5$).

The second line contains $\frac{n}{2}$ positive integers $x_2, x_4, \ldots, x_n$ ($1 \le x_i \le 2 \cdot 10^5$).

Output

If there are no possible sequence, print "No".

Otherwise, print "Yes" and then n positive integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^{13}$), where $x_2, x_4, \ldots, x_n$ should be same as in input data. If there are multiple answers, print any.

Note, that the limit for x_i is larger than for input data. It can be proved that in case there is an answer, there must be a possible sequence satisfying $1 \le x_i \le 10^{13}$.

Examples

input
6 5 11 44
output
Yes 4 5 16 11 64 44

input
2 9900
output
Yes 100 9900

input
6 314 1592 6535
output
No

Note

In the first example

- $x_1 = 4$
- $x_1 + x_2 = 9$
- $x_1 + x_2 + x_3 = 25$
- $x_1 + x_2 + x_3 + x_4 = 36$
- $x_1 + x_2 + x_3 + x_4 + x_5 = 100$
- $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 144$

All these numbers are perfect squares.
In the second example, $x_1 = 100$, $x_1 + x_2 = 10000$. They are all perfect squares. There're other answers possible. For example, $x_1 = 22500$ is another answer.

In the third example, it is possible to show, that no such sequence exists.

F. Tricky Interactor

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

This is an interactive problem.

Chouti was tired of studying, so he opened the computer and started playing a puzzle game.

Long long ago, the boy found a sequence $s_1, s_2, \ldots, s_n$ of length n, kept by a tricky interactor. It consisted of 0s and 1s only and the number of 1s is t. The boy knows nothing about this sequence except n and t, but he can try to find it out with some queries with the interactor.

We define an operation called flipping. Flipping $[l, r]$ ($1 \le l \le r \le n$) means for each $x \in [l, r]$, changing s_x to $1 - s_x$.

In each query, the boy can give the interactor two integers l, r satisfying $1 \le l \le r \le n$ and the interactor will either flip $[1, r]$ or $[l, n]$ (both outcomes have the same probability and all decisions made by interactor are independent from each other, see `Notes` section for more details). The interactor will tell the current number of 1s after each operation. Note, that the sequence **won't be restored** after each operation.

Help the boy to find the **original** sequence in no more than 10000 interactions.

"Weird legend, dumb game." he thought. However, after several tries, he is still stuck with it. Can you help him beat this game?

Interaction

The interaction starts with a line containing two integers n and t ($1 \le n \le 300, 0 \le t \le n$) — the length of the sequence and the number of 1s in it.

After that, you can make queries.

To make a query, print a line "`? l r`" ($1 \le l \le r \le n$), then `flush` the output.

After each query, read a single integer t ($-1 \le t \le n$).

- If $t = -1$, it means that you're out of queries, you should terminate your program immediately, then you will get `Wrong Answer`, otherwise the judging result would be undefined because you will interact with a closed stream.
- If $t \ge 0$, it represents the current number of 1s in the sequence.

When you found the original sequence, print a line "`! s`", `flush` the output and terminate. Print $s_1, s_2, \ldots, s_n$ as a binary string and do not print spaces in between.

Your solution will get `Idleness Limit Exceeded` if you don't print anything or forget to flush the output.

To `flush` you need to do the following right after printing a query and a line end:

- `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

For hacks, use the following format:

In the first and only line, print a non-empty binary string. Its length will be n and it will be treated as $s_1, s_2, \ldots, s_n$.

For example, the the test corresponding to the example contains a line "`0011`".

Example

input
4 2 2 2 0
output
? 1 1 ? 1 1 ? 3 4 ! 0011

Note

For first query $1, 1$, the interactor should flip $[1, 1]$ or $[1, 4]$. It chose to flip $[1, 4]$, so the sequence became `1100`.

For second query $1, 1$, the interactor should flip $[1, 1]$ or $[1, 4]$. It again chose to flip $[1, 4]$, so the sequence became `0011`.

For third query $3, 4$, the interactor should flip $[1, 4]$ or $[3, 4]$. It chose to flip $[3, 4]$, so the sequence became `0000`.

Q: How does interactor choose between $[1, r]$ and $[l, n]$? Is it really random?

A: The interactor will use a **secret** pseudorandom number generator. Only s and your queries will be hashed and used as the seed. So if you give the same sequence of queries twice for the same secret string, you will get the same results. Except this, you can consider the choices fully random, like flipping a fair coin. **You needn't (and shouldn't) exploit the exact generator to pass this problem.**

G. Mergesort Strikes Back

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

Chouti thought about his very first days in competitive programming. When he had just learned to write merge sort, he thought that the merge sort is too slow, so he restricted the maximum depth of recursion and modified the merge sort to the following:

Function — a fake MergeSort function

```
1:  function MergeSort(a, l, r, h)
2:      b ← an empty array
3:      if l ≤ r then
4:          if h ≤ 1 then                ▷ the depth reached the fix threshold
5:              for i ← l to r do
6:                  append the i-th element of a to the end of b
7:          else                                          ▷ split and recursion
8:              m ← ⌊(l + r)/2⌋
9:              c ← MergeSort(a, l, m, h − 1)
10:             d ← MergeSort(a, m + 1, r, h − 1)
11:                                                  ▷ merge two arrays
12:             while c is not empty and d is not empty do
13:                 x ← the front element of c
14:                 y ← the front element of d
15:                 if x < y then
16:                     append x to the end of b
17:                     remove the front element of c
18:                 else
19:                     append y to the end of b
20:                     remove the front element of d
21:             if c is not empty then
22:                 append all the elements of c to the end of b
23:             if d is not empty then
24:                 append all the elements of d to the end of b
25:     return b
```

Chouti found his idea dumb since obviously, this "merge sort" sometimes cannot sort the array correctly. However, Chouti is now starting to think of how good this "merge sort" is. Particularly, Chouti wants to know for a random permutation a of $1, 2, \ldots, n$ the expected number of inversions after calling `MergeSort(a, 1, n, k)`.

It can be proved that the expected number is rational. For the given prime q, suppose the answer can be denoted by $\frac{u}{d}$ where $gcd(u, d) = 1$, you need to output an integer r satisfying $0 \le r < q$ and $rd \equiv u \pmod{q}$. It can be proved that such r exists and is unique.

Input

The first and only line contains three integers n, k, q ($1 \le n, k \le 10^5, 10^8 \le q \le 10^9$, q is a prime).

Output

The first and only line contains an integer r.

Examples

input
3 1 998244353
output
499122178

input
3 2 998244353
output
665496236

input
9 3 998244353
output
449209967

input
9 4 998244353
output
665496237

Note

In the first example, all possible permutations are $[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1]$.

With $k = 1$, `MergeSort(a, 1, n, k)` will only return the original permutation. Thus the answer is $9/6 = 3/2$, and you should output 499122178 because $499122178 \times 2 \equiv 3 \pmod{998244353}$.

In the second example, all possible permutations are $[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1]$ and the corresponding outputs of `MergeSort(a, 1, n, k)` are $[1,2,3], [1,2,3], [2,1,3], [1,2,3], [2,3,1], [1,3,2]$ respectively. Thus the answer is $4/6 = 2/3$, and you should output 665496236 because $665496236 \times 3 \equiv 2 \pmod{998244353}$.

H. Palindromic Magic

time limit per test: 2.5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

After learning some fancy algorithms about palindromes, Chouti found palindromes very interesting, so he wants to challenge you with this problem.

Chouti has got two strings A and B. Since he likes palindromes, he would like to pick a as some non-empty palindromic substring of A and b as some non-empty palindromic substring of B. Concatenating them, he will get string ab.

Chouti thinks strings he could get this way are interesting, so he wants to know how many different strings he can get.

Input

The first line contains a single string A ($1 \leq |A| \leq 2 \cdot 10^5$).

The second line contains a single string B ($1 \leq |B| \leq 2 \cdot 10^5$).

Strings A and B contain only lowercase English letters.

Output

The first and only line should contain a single integer — the number of possible strings.

Examples

input
aa aba
output
6

input
aaba abaa
output
15

Note

In the first example, attainable strings are

- "a" + "a" = "aa",
- "aa" + "a" = "aaa",
- "aa" + "aba" = "aaaba",
- "aa" + "b" = "aab",
- "a" + "aba" = "aaba",
- "a" + "b" = "ab".

In the second example, attainable strings are "aa", "aaa", "aaaa", "aaaba", "aab", "aaba", "ab", "abaa", "abaaa", "abaaba", "abab", "ba", "baa", "baba", "bb".

Notice that though "a"+"aa"="aa"+"a"="aaa", "aaa" will only be counted once.
