A. Strange Addition

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

Unfortunately, Vasya can only sum pairs of integers (a, b), such that for any decimal place at least one number has digit 0 in this place. For example, Vasya can sum numbers 505 and 50, but he cannot sum 1 and 4.

Vasya has a set of k distinct non-negative integers $d_1, d_2, ..., d_k$.

Vasya wants to choose some integers from this set so that he could sum any two chosen numbers. What maximal number of integers can he choose in the required manner?

Input

The first input line contains integer k $(1 \le k \le 100)$ — the number of integers.

The second line contains k distinct space-separated integers $d_1, d_2, ..., d_k$ $(0 \le d_i \le 100)$.

Output

In the first line print a single integer n the maximum number of the chosen integers. In the second line print n distinct non-negative integers — the required integers.

If there are multiple solutions, print any of them. You can print the numbers in any order.

Sample test(s)

input

```
4
100 10 1 0
```

output

```
4
0 1 10 100
```

input

```
3
2 70 3
```

output

```
2
2 70
```

B. Continued Fractions

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

A continued fraction of height n is a fraction of form $a_1 + \frac{1}{a_2 + \frac{1}{\dots + \frac{1}{a_n}}}$. You are given two rational numbers, one is represented as $\frac{p}{q}$ and the other one is represented as a finite fraction of height n. Check if they are equal.

Input

The first line contains two space-separated integers p, q $(1 \le q \le p \le 10^{18})$ — the numerator and the denominator of the first fraction.

The second line contains integer n $(1 \le n \le 90)$ — the height of the second fraction. The third line contains n space-separated integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^{18})$ — the continued fraction.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print "`YES`" if these fractions are equal and "`NO`" otherwise.

Sample test(s)

input

```
9 4
2
2 4
```

output

```
YES
```

input

```
9 4
3
2 3 1
```

output

```
YES
```

input

```
9 4
3
1 2 4
```

output

```
NO
```

Note

In the first sample $2 + \frac{1}{4} = \frac{9}{4}$.

In the second sample $2 + \frac{1}{3 + \frac{1}{1}} = 2 + \frac{1}{4} = \frac{9}{4}$.

In the third sample $1 + \frac{1}{2 + \frac{1}{4}} = \frac{13}{9}$.

C. Ivan and Powers of Two

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

Ivan has got an array of n non-negative integers $a_1, a_2, ..., a_n$. Ivan knows that the array is sorted in the non-decreasing order.

Ivan wrote out integers $2^{a_1}, 2^{a_2}, ..., 2^{a_n}$ on a piece of paper. Now he wonders, what minimum number of integers of form 2^b $(b \ge 0)$ need to be added to the piece of paper so that the sum of all integers written on the paper equalled $2^v - 1$ for some integer v $(v \ge 0)$.

Help Ivan, find the required quantity of numbers.

Input

The first line contains integer n $(1 \le n \le 10^5)$. The second input line contains n space-separated integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 2 \cdot 10^9)$. It is guaranteed that $a_1 \le a_2 \le ... \le a_n$.

Output

Print a single integer — the answer to the problem.

Sample test(s)

input

```
4
0 1 1 1
```

output

```
0
```

input

```
1
3
```

output

```
3
```

Note

In the first sample you do not need to add anything, the sum of numbers already equals $2^3 - 1 = 7$.

In the second sample you need to add numbers $2^0, 2^1, 2^2$.

D. Olya and Graph

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

Olya has got a directed non-weighted graph, consisting of n vertexes and m edges. We will consider that the graph vertexes are indexed from 1 to n in some manner. Then for any graph edge that goes from vertex v to vertex u the following inequation holds: $v < u$.

Now Olya wonders, how many ways there are to add an arbitrary (possibly zero) number of edges to the graph so as the following conditions were met:

1. You can reach vertexes number $i + 1, i + 2, ..., n$ from any vertex number i $(i < n)$.
2. For any graph edge going from vertex v to vertex u the following inequation fulfills: $v < u$.
3. There is at most one edge between any two vertexes.
4. The shortest distance between the pair of vertexes i, j $(i < j)$, for which $j - i \le k$ holds, equals $j - i$ edges.
5. The shortest distance between the pair of vertexes i, j $(i < j)$, for which $j - i > k$ holds, equals either $j - i$ or $j - i - k$ edges.

We will consider two ways *distinct*, if there is the pair of vertexes i, j $(i < j)$, such that first resulting graph has an edge from i to j and the second one doesn't have it.

Help Olya. As the required number of ways can be rather large, print it modulo 1000000007 $(10^9 + 7)$.

Input

The first line contains three space-separated integers n, m, k $(2 \le n \le 10^6, 0 \le m \le 10^5, 1 \le k \le 10^6)$.

The next m lines contain the description of the edges of the initial graph. The i-th line contains a pair of space-separated integers u_i, v_i $(1 \le u_i < v_i \le n)$ — the numbers of vertexes that have a directed edge from u_i to v_i between them.

It is guaranteed that any pair of vertexes u_i, v_i has at most one edge between them. It also is guaranteed that the graph edges are given in the order of non-decreasing u_i. If there are multiple edges going from vertex u_i, then it is guaranteed that these edges are given in the order of increasing v_i.

Output

Print a single integer — the answer to the problem modulo 1000000007 $(10^9 + 7)$.

Sample test(s)

input
```
7 8 2
1 2
2 3
3 4
3 6
4 5
4 7
5 6
6 7
```
output
```
2
```

input
```
7 0 2
```
output
```
12
```

input
```
7 2 1
1 3
3 5
```
output
```
0
```

Note

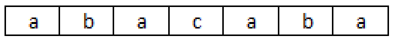
In the first sample there are two ways: the first way is not to add anything, the second way is to add a single edge from vertex 2 to vertex 5.

E. Playing with String

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

Two people play the following string game. Initially the players have got some string s. The players move in turns, the player who cannot make a move loses.

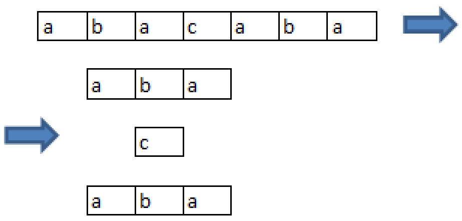
Before the game began, the string is written on a piece of paper, one letter per cell.



An example of the initial situation at s = "abacaba"

A player's move is the sequence of actions:

1. The player chooses one of the available pieces of paper with some string written on it. Let's denote it is t. Note that initially, only one piece of paper is available.
2. The player chooses in the string $t = t_1t_2... t_{|t|}$ character in position i $(1 \le i \le |t|)$ such that for some positive integer l $(0 < i - l;\ i + l \le |t|)$ the following equations hold: $t_{i-1} = t_{i+1}$, $t_{i-2} = t_{i+2}$, ..., $t_{i-l} = t_{i+l}$.
3. Player cuts the cell with the chosen character. As a result of the operation, he gets three new pieces of paper, the first one will contain string $t_1t_2... t_{i-1}$, the second one will contain a string consisting of a single character t_i, the third one contains string $t_{i+1}t_{i+2}... t_{|t|}$.



An example of making action $(i = 4)$ with string s = «abacaba»

Your task is to determine the winner provided that both players play optimally well. If the first player wins, find the position of character that is optimal to cut in his first move. If there are multiple positions, print the minimal possible one.

Input

The first line contains string s $(1 \le |s| \le 5000)$. It is guaranteed that string s only contains lowercase English letters.

Output

If the second player wins, print in the single line "Second" (without the quotes). Otherwise, print in the first line "First" (without the quotes), and in the second line print the minimal possible winning move — integer i $(1 \le i \le |s|)$.

Sample test(s)

input

```
abacaba
```

output

```
First
2
```

input

```
abcde
```

output

```
Second
```

Note

In the first sample the first player has multiple winning moves. But the minimum one is to cut the character in position 2.

In the second sample the first player has no available moves.

Codeforces (c) Copyright 2010-2015 Mike Mirzayanov
The only programming contests Web 2.0 platform