A. BowWow and the Timetable

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

In the city of Saint Petersburg, a day lasts for 2^{100} minutes. From the main station of Saint Petersburg, a train departs after 1 minute, 4 minutes, 16 minutes, and so on; in other words, the train departs at time 4^k for each integer $k \geq 0$. Team BowWow has arrived at the station at the time s and it is trying to count how many trains have they missed; in other words, the number of trains that have departed **strictly before** time s. For example if $s = 20$, then they missed trains which have departed at 1, 4 and 16. As you are the only one who knows the time, help them!

Note that the number s will be given you in a binary representation without leading zeroes.

Input

The first line contains a single **binary number** s ($0 \leq s < 2^{100}$) without leading zeroes.

Output

Output a single number — the number of trains which have departed strictly before the time s.

Examples

input
```
100000000
```
output
```
4
```

input
```
101
```
output
```
2
```

input
```
10100
```
output
```
3
```

Note

In the first example $100000000_2 = 256_{10}$, missed trains have departed at 1, 4, 16 and 64.

In the second example $101_2 = 5_{10}$, trains have departed at 1 and 4.

The third example is explained in the statements.

B. Mislove Has Lost an Array

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

Mislove had an array $a_1, a_2, \cdots, a_n$ of n positive integers, but he has lost it. He only remembers the following facts about it:

- The number of different numbers in the array is not less than l and is not greater than r;
- For each array's element a_i either $a_i = 1$ or a_i is even and there is a number $\dfrac{a_i}{2}$ in the array.

For example, if $n = 5$, $l = 2$, $r = 3$ then an array could be $[1, 2, 2, 4, 4]$ or $[1, 1, 1, 1, 2]$; but it couldn't be $[1, 2, 2, 4, 8]$ because this array contains 4 different numbers; it couldn't be $[1, 2, 2, 3, 3]$ because 3 is odd and isn't equal to 1; and it couldn't be $[1, 1, 2, 2, 16]$ because there is a number 16 in the array but there isn't a number $\frac{16}{2} = 8$.

According to these facts, he is asking you to count the minimal and the maximal possible sums of all elements in an array.

Input

The only input line contains three integers n, l and r ($1 \le n \le 1\,000$, $1 \le l \le r \le \min(n, 20)$) — an array's size, the minimal number and the maximal number of distinct elements in an array.

Output

Output two numbers — the minimal and the maximal possible sums of all elements in an array.

Examples

input

```
4 2 2
```

output

```
5 7
```

input

```
5 1 5
```

output

```
5 31
```

Note

In the first example, an array could be the one of the following: $[1, 1, 1, 2]$, $[1, 1, 2, 2]$ or $[1, 2, 2, 2]$. In the first case the minimal sum is reached and in the last case the maximal sum is reached.

In the second example, the minimal sum is reached at the array $[1, 1, 1, 1, 1]$, and the maximal one is reached at the array $[1, 2, 4, 8, 16]$.

C. Anna, Svyatoslav and Maps

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

The main characters have been omitted to be short.

You are given a directed unweighted graph without loops with n vertexes and a path in it (that path is not necessary simple) given by a sequence $p_1, p_2, \ldots, p_m$ of m vertexes; for each $1 \le i < m$ there is an arc from p_i to p_{i+1}.

Define the sequence $v_1, v_2, \ldots, v_k$ of k vertexes as *good*, if v is a subsequence of p, $v_1 = p_1$, $v_k = p_m$, and p is one of the shortest paths passing through the vertexes v_1, …, v_k in that order.

A sequence a is a subsequence of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements. It is obvious that the sequence p is good but your task is to find the **shortest** good subsequence.

If there are multiple shortest good subsequences, output any of them.

Input

The first line contains a single integer n ($2 \le n \le 100$) — the number of vertexes in a graph.

The next n lines define the graph by an adjacency matrix: the j-th character in the i-st line is equal to 1 if there is an arc from vertex i to the vertex j else it is equal to 0. It is guaranteed that the graph doesn't contain loops.

The next line contains a single integer m ($2 \le m \le 10^6$) — the number of vertexes in the path.

The next line contains m integers $p_1, p_2, \ldots, p_m$ ($1 \le p_i \le n$) — the sequence of vertexes in the path. It is guaranteed that for any $1 \le i < m$ there is an arc from p_i to p_{i+1}.

Output

In the first line output a single integer k ($2 \le k \le m$) — the length of the shortest good subsequence. In the second line output k integers v_1, …, v_k ($1 \le v_i \le n$) — the vertexes in the subsequence. If there are multiple shortest subsequences, print any. Any two consecutive numbers should be distinct.

Examples

input

```
4
0110
0010
0001
1000
4
1 2 3 4
```

output

```
3
1 2 4
```

input

```
4
0110
0010
1001
1000
20
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

output

```
11
1 2 4 2 4 2 4 2 4 2 4
```

input

```
3
011
101
110
7
1 2 3 1 3 2 1
```

output

```
7
1 2 3 1 3 2 1
```

input

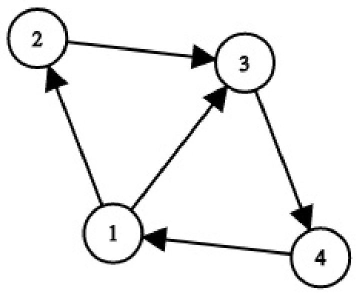
```
4
0110
0001
0001
1000
3
1 2 4
```

output

```
2
1 4
```

Note

Below you can see the graph from the first example:



The given path is passing through vertexes $1, 2, 3, 4$. The sequence $1-2-4$ is good because it is the subsequence of the given path, its first and the last elements are equal to the first and the last elements of the given path respectively, and the shortest path passing through vertexes $1, 2$ and 4 in that order is $1-2-3-4$. Note that subsequences $1-4$ and $1-3-4$ aren't good because in both cases the shortest path passing through the vertexes of these sequences is $1-3-4$.

In the third example, the graph is full so any sequence of vertexes in which any two consecutive elements are distinct defines a path consisting of the same number of vertexes.

In the fourth example, the paths $1-2-4$ and $1-3-4$ are the shortest paths passing through the vertexes 1 and 4.

D1. Kirk and a Binary String (easy version)

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

The only difference between easy and hard versions is the length of the string. You can hack this problem only if you solve both problems.

Kirk has a binary string s (a string which consists of zeroes and ones) of length n and he is asking you to find a binary string t of the same length which satisfies the following conditions:

- For any l and r ($1 \leq l \leq r \leq n$) the length of the longest non-decreasing subsequence of the substring $s_l s_{l+1} \ldots s_r$ is equal to the length of the longest non-decreasing subsequence of the substring $t_l t_{l+1} \ldots t_r$;

- The number of zeroes in t is the maximum possible.

A non-decreasing subsequence of a string p is a sequence of indices $i_1, i_2, \ldots, i_k$ such that $i_1 < i_2 < \ldots < i_k$ and $p_{i_1} \leq p_{i_2} \leq \ldots \leq p_{i_k}$. The length of the subsequence is k.

If there are multiple substrings which satisfy the conditions, output any.

Input

The first line contains a binary string of length not more than $2\,000$.

Output

Output a binary string which satisfied the above conditions. If there are many such strings, output any of them.

Examples

input
```
110
```
output
```
010
```

input
```
010
```
output
```
010
```

input
```
0001111
```
output
```
0000000
```

input
```
0111001100111011101000
```
output
```
0011001100001011101000
```

Note

In the first example:

- For the substrings of the length 1 the length of the longest non-decreasing subsequnce is 1;
- For $l = 1, r = 2$ the longest non-decreasing subsequnce of the substring s_1s_2 is 11 and the longest non-decreasing subsequnce of the substring t_1t_2 is 01;
- For $l = 1, r = 3$ the longest non-decreasing subsequnce of the substring s_1s_3 is 11 and the longest non-decreasing subsequnce of the substring t_1t_3 is 00;
- For $l = 2, r = 3$ the longest non-decreasing subsequnce of the substring s_2s_3 is 1 and the longest non-decreasing subsequnce of the substring t_2t_3 is 1;

The second example is similar to the first one.

D2. Kirk and a Binary String (hard version)

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

The only difference between easy and hard versions is the length of the string. You can hack this problem if you solve it. But you can hack the previous problem only if you solve both problems.

Kirk has a binary string s (a string which consists of zeroes and ones) of length n and he is asking you to find a binary string t of the same length which satisfies the following conditions:

- For any l and r ($1 \leq l \leq r \leq n$) the length of the longest non-decreasing subsequence of the substring $s_ls_{l+1}\ldots s_r$ is equal to the length of the longest non-decreasing subsequence of the substring $t_lt_{l+1}\ldots t_r$;
- The number of zeroes in t is the maximum possible.

A non-decreasing subsequence of a string p is a sequence of indices $i_1, i_2, \ldots, i_k$ such that $i_1 < i_2 < \ldots < i_k$ and $p_{i_1} \leq p_{i_2} \leq \ldots \leq p_{i_k}$. The length of the subsequence is k.

If there are multiple substrings which satisfy the conditions, output any.

Input

The first line contains a binary string of length not more than 10^5.

Output

Output a binary string which satisfied the above conditions. If there are many such strings, output any of them.

Examples

input

```
110
```

output

```
010
```

input

```
010
```

output

```
010
```

input

```
0001111
```

output

```
0000000
```

input

```
0111001100111011101000
```

output

```
0011001100001011101000
```

Note

In the first example:

- For the substrings of the length 1 the length of the longest non-decreasing subsequnce is 1;
- For $l = 1, r = 2$ the longest non-decreasing subsequnce of the substring $s_1 s_2$ is 11 and the longest non-decreasing subsequnce of the substring $t_1 t_2$ is 01;
- For $l = 1, r = 3$ the longest non-decreasing subsequnce of the substring $s_1 s_3$ is 11 and the longest non-decreasing subsequnce of the substring $t_1 t_3$ is 00;
- For $l = 2, r = 3$ the longest non-decreasing subsequnce of the substring $s_2 s_3$ is 1 and the longest non-decreasing subsequnce of the substring $t_2 t_3$ is 1;

The second example is similar to the first one.

E. Natasha, Sasha and the Prefix Sums

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

Natasha's favourite numbers are n and 1, and Sasha's favourite numbers are m and -1. One day Natasha and Sasha met and wrote down every possible array of length $n + m$ such that some n of its elements are equal to 1 and another m elements are equal to -1. For each such array they counted its maximal prefix sum, probably an empty one which is equal to 0 (in another words, if every nonempty prefix sum is less to zero, then it is considered equal to zero). Formally, denote as $f(a)$ the maximal prefix sum of an array $a_{1,\ldots,l}$ of length $l \geq 0$. Then:

$$f(a) = \max(0, \max_{1 \leq i \leq l} \sum_{j=1}^{i} a_j)$$

Now they want to count the sum of maximal prefix sums for each such an array and they are asking you to help. As this sum can be very large, output it modulo $998\,244\,853$.

Input

The only line contains two integers n and m ($0 \leq n, m \leq 2\,000$).

Output

Output the answer to the problem modulo $998\,244\,853$.

Examples

input

```
0 2
```

output

```
0
```

input

```
2 0
```

output

```
2
```

input

```
2 2
```

output

```
5
```

input

```
2000 2000
```

output

```
674532367
```

Note

In the first example the only possible array is `[-1,-1]`, its maximal prefix sum is equal to 0.

In the second example the only possible array is `[1,1]`, its maximal prefix sum is equal to 2.

There are 6 possible arrays in the third example:

`[1,1,-1,-1]`, `f([1,1,-1,-1]) = 2`

`[1,-1,1,-1]`, `f([1,-1,1,-1]) = 1`

`[1,-1,-1,1]`, `f([1,-1,-1,1]) = 1`

`[-1,1,1,-1]`, `f([-1,1,1,-1]) = 1`

`[-1,1,-1,1]`, `f([-1,1,-1,1]) = 0`

`[-1,-1,1,1]`, `f([-1,-1,1,1]) = 0`

So the answer for the third example is $2+1+1+1+0+0=5$.