Maximum Prime

We all know that computers understand nothing but binary language, which consists of only zeros and ones.

There is a task assigned to you. Given a binary string, the subsequences of this string can be converted into numbers. The task is about finding the maximum prime number of these numbers if possible.

Note: A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements

Example

Given a binary string *binary_string* = "1011". The subsequences of this string are:

{"1", "10", "101", "1011", "101", "10", "101", "10", "1", "11", "111", "11", "1", "11", "1", "0", "01", "011", "01", "0", "01", "0", "1", "11"

respectively.

- The second line has a string of length n representing s.

Output format

Print the lexicographic smallest possible subsequence of length k.

Constraints

$1 \leq n \leq 10^5$
$1 \leq x \leq k \leq n$
$x <= Number\ of\ 'b'\ in\ s$

Sample input	Sample output
4 3 2 baba	bab

- s: Represents a string denoting s.

Input format

Note: This is the input format that you must use to provide custom input (available above the **Compile and Test** button).

- The first line has an three space separated integers n , k and x denoting the length of input string, output string ,minimum required 'b' respectively.
- The second line has a string of length n representing s.

Output format

Print the lexicographic smallest possible subsequence of length k.

Constraints

$$1 \leq n \leq 10^5$$
$$1 \leq x \leq k \leq n$$
$$x <= Number\ of\ 'b'\ in$$

It is guaranteed that answer always exist.

3

Example

Let's assume that n is 6, k is 4 ,x is 2 and s is *aababb*.

Now the subsequence of s of length 4 that has atleast 2 b's are { aabb, abab, abbb, babb }.

We can see that the lexographically smallest among them is aabb, which is the required answer.

Function description

Complete the *solve* function provided in the editor. This function takes the following 4 parameters and returns the **lexicographically smallest** possible **subsequence** of s of **length** k having **atleast** x **'b'**.

- n: Represents an integer denoting the length of the input string
- k: Represents an integer denoting the length of

Question 3 Max. score: 20.00

1 A Beautiful Sequence!

2 You are given a string $s = s_1 s_2 \cdots s_n$ of length ,

3 which only contains of letters a and b.

Find the **lexicographically smallest** possible **subsequence** of s of **length** k having **atleast** x $'b'$.

String A is lexicographically smaller than string B if it is shorter than B $(|A| < |B|)$ and is its prefix, or if none of them is a prefix of the other and at the first position where they differ character in A is smaller than the character in B.

A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements without changing the order of the remaining elements.

Note

It is guaranteed that answer always exist.

2

3

Explanation

The maximum prime number is the integer value of the subsequence "101" which is equal to 5.

Note:

Your code must be able to print the sample output from the provided sample input. However, your code is run against multiple hidden test cases. Therefore, your code must pass these hidden test cases to solve the problem statement.

Limits

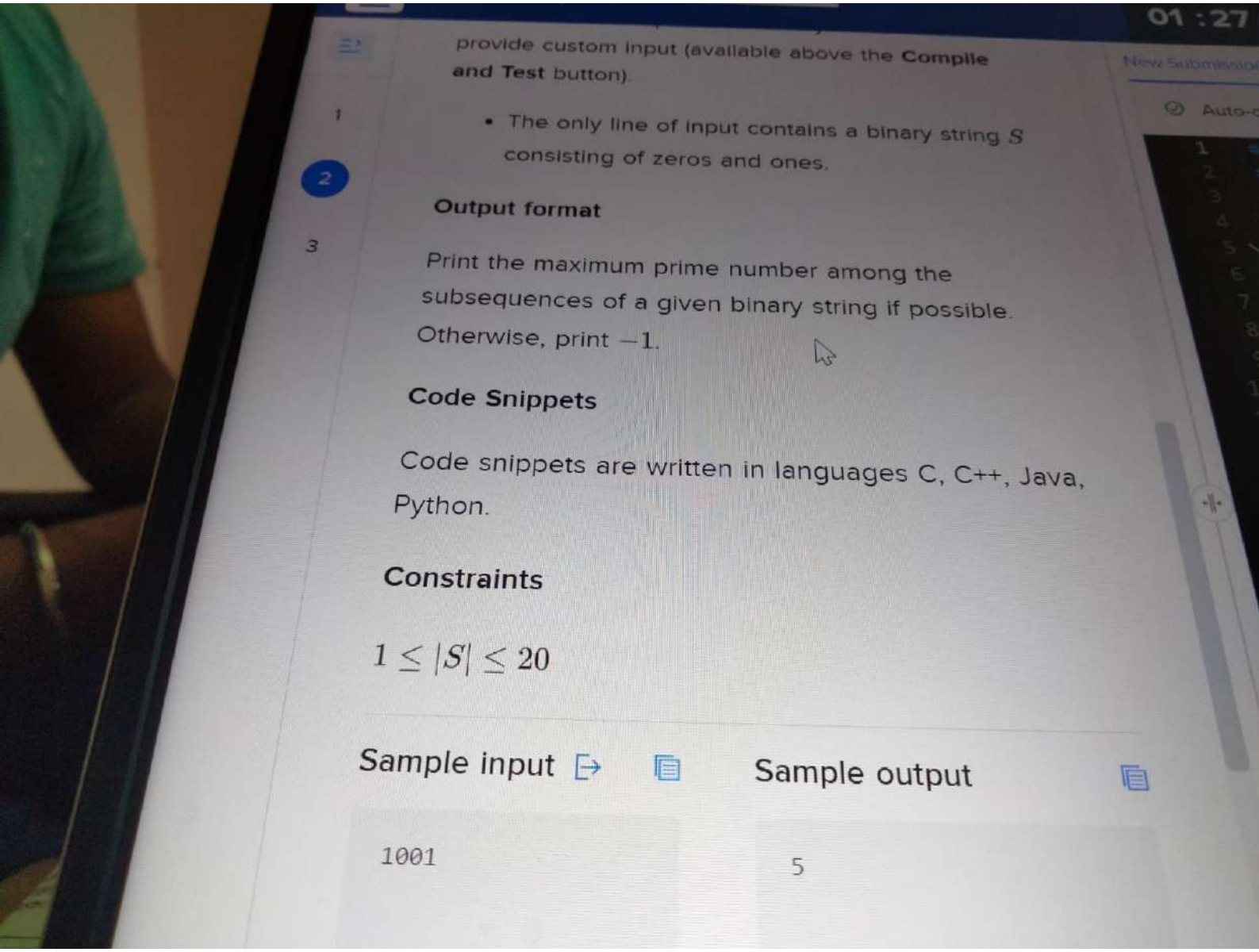
Time Limit: 0.5 sec(s) for each input file
Memory Limit: 256 MB
Source Limit: 1024 KB

Scoring

Score is assigned if any testcase passes

Allowed Languages

Bash, C, C++14, C++17, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java 8, Java 14, JavaScript(Node.js), Julia, Kotlin, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl


01 :27
provide custom input (available above the Compile and Test button).
The only line of input contains a binary string S consisting of zeros and ones.
Output format
Print the maximum prime number among the subsequences of a given binary string if possible. Otherwise, print -1.
Code Snippets
Code snippets are written in languages C, C++, Java, Python.
Constraints
$1 \leq |S| \leq 20$
Sample input
1001
Sample output
5

subsequences of this string are:

{"1", "10", "101", "1011", "101", "10", "101", "10", "1", "11", "111", "11", "1", "11", "1", "0", "01", "011", "01", "0", "01", "0", "1", "11", "1", "1"}.

So the maximum prime number among these subsequences is "111" which equal to 7.

Function description

Complete the *solve* function provided in the editor. This function takes the following parameter and returns the maximum prime number among the subsequences of the given binary string if possible or return -1.

- S: Represents the binary string.

Input format

Note: This is the input format that you must use to provide custom input (available above the **Compile and Test** button).

- The only line of input contains a binary string S

reach the bottom right corner using either glasses, print -1.

Code Snippets

Code snippets are written in languages C, C++, Java, Python.

Constraints

- $1 \leq N, M \leq 500$
- It is guaranteed that all values of the maze consist of '.' for paths, '*' for walls, and '#' for doors.
- The top left and bottom right corners of the maze will not be obstacles.

Sample input

```
5 5
....*
****.
.....
.##.#
```

Sample output

```
-1
8
```

maze.

Input format

- The first line contains two integers, *N*, *M* denoting the number of rows and columns in the grid.
- The following *rows* lines will have strings of the same length representing the maze. Each character in *maze[i]* will be either a '.' representing a clear passage or an '*' representing a wall or an '#' representing a door.

Output format

Two lines contain two integers, the first one denotes the minimum time you need to reach the bottom right corner with the first glasses, the second entity denotes the minimum time you need to reach the destination using the second glasses. If you can't reach the bottom right corner using either glasses, print -1.

2 If you use the second glass can reach the bottom right corner following

3 $(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,4) \rightarrow (2,4) \rightarrow (3,4) \rightarrow (4,4)$ ie by using 7 steps.

Function description

Complete the *solve* function provided in the editor. This function takes the following 2 parameters and returns an integer array of length 2 containing minimum time required by each glasses.

- *N*: Represents an integer denoting the number of rows in the grid.
- *M*: Represents an integer denoting the number of columns in the grid.
- *maze*: Represents a string array of size *N* , in which each string denotes each row of the maze.

represented in doors and walls. Each type of glasses allows passing one type of obstacle but not the other one.

How long do you need to reach the bottom right corner using the first glasses and second glasses?

Given the grid which represents the maze. Print the minimum time needed to reach the bottom right corner using the first and second glasses. If you can't reach the bottom right corner using either glasses, print -1.

Example

Let's assume $N = 4$ and $M = 5$, and the grid is as follows

```
. . . . .
* * . * *
. # # . #
. . . . .
```

represented in doors and walls. Each type of glasses allows passing one type of obstacle but not the other one.

How long do you need to reach the bottom right corner using the first glasses and second glasses?

Given the grid which represents the maze. Print the minimum time needed to reach the bottom right corner using the first and second glasses. If you can't reach the bottom right corner using either glasses, print -1.

Example

Let's assume $N = 4$ and $M = 5$, and the grid is as follows

```
. . . . .
* * . * *
. # # . #
. . . . .
```

1

The Maze Runner

2

You bought two virtual reality glasses. There is only one game installed to both of them called "The Maze Runner".

3

Each glasses has its special feature.

1. The first glasses, allows you to pass through doors.
2. The second glasses, allows you to walk through walls.

The maze looks like a gird of size $N \times M$. At first, you are at the top left corner of this grid. You want to reach the bottom right corner of this maze. You can move vertically or horizontally on the grid and need 1 minute to go from one cell to another. The cells with

reach the bottom right corner using either glasses, print -1.

Code Snippets

Code snippets are written in languages C, C++, Java, Python.

Constraints

- $1 \leq N, M \leq 500$
- It is guaranteed that all values of the maze consist of '.' for paths, '*' for walls, and '#' for doors.
- The top left and bottom right corners of the maze will not be obstacles.

Sample input

```
5 5
....*
****.
.....
.##.#
```

Sample output

```
-1
8
```

maze.

Input format

- The first line contains two integers, *N*, *M* denoting the number of rows and columns in the grid.
- The following *rows* lines will have strings of the same length representing the maze. Each character in *maze[i]* will be either a '.' representing a clear passage or an '*' representing a wall or an '#' representing a door.

Output format

Two lines contain two integers, the first one denotes the minimum time you need to reach the bottom right corner with the first glasses, the second entity denotes the minimum time you need to reach the destination using the second glasses. If you can't reach the bottom right corner using either glasses, print -1.

2 If you use the second glass can reach the bottom right corner following

3 $(1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,4) \rightarrow (2,4) \rightarrow (3,4) \rightarrow (4,4)$ ie by using 7 steps.

Function description

Complete the *solve* function provided in the editor. This function takes the following 2 parameters and returns an integer array of length 2 containing minimum time required by each glasses.

- *N*: Represents an integer denoting the number of rows in the grid.
- *M*: Represents an integer denoting the number of columns in the grid.
- *maze*: Represents a string array of size *N* , in which each string denotes each row of the maze.

represented in doors and walls. Each type of glasses allows passing one type of obstacle but not the other one.

How long do you need to reach the bottom right corner using the first glasses and second glasses?

Given the grid which represents the maze. Print the minimum time needed to reach the bottom right corner using the first and second glasses. If you can't reach the bottom right corner using either glasses, print -1.

Example

Let's assume $N = 4$ and $M = 5$, and the grid is as follows

```
. . . . .
* * . * *
. # # . #
. . . . .
```

1

The Maze Runner

2

You bought two virtual reality glasses. There is only one game installed to both of them called "The Maze Runner".

3

Each glasses has its special feature.

1. The first glasses, allows you to pass through doors.
2. The second glasses, allows you to walk through walls.

The maze looks like a gird of size $N \times M$. At first, you are at the top left corner of this grid. You want to reach the bottom right corner of this maze. You can move vertically or horizontally on the grid and need 1 minute to go from one cell to another. The cells with

respectively.

- The second line has a string of length n representing s.

Output format

Print the lexicographic smallest possible subsequence of length k.

Constraints

$1 \leq n \leq 10^5$
$1 \leq x \leq k \leq n$
$x <= Number\ of\ 'b'\ in\ s$

Sample input	Sample output
4 3 2 baba	bab

It is guaranteed that answer always exist.

3

Example

Let's assume that n is 6, k is 4 ,x is 2 and s is *aababb*.

Now the subsequence of s of length 4 that has atleast 2 b's are { aabb, abab, abbb, babb }.

We can see that the lexographically smallest among them is aabb, which is the required answer.

Function description

Complete the *solve* function provided in the editor. This function takes the following 4 parameters and returns the **lexicographically smallest** possible **subsequence** of s of **length** k having **atleast** x **'b'**.

- n: Represents an integer denoting the length of the input string
- k: Represents an integer denoting the length of

- s: Represents a string denoting s.

Input format

Note: This is the input format that you must use to provide custom input (available above the **Compile and Test** button).

- The first line has an three space separated integers n , k and x denoting the length of input string, output string ,minimum required 'b' respectively.
- The second line has a string of length n representing s.

Output format

Print the lexicographic smallest possible subsequence of length k.

Constraints

$$1 \leq n \leq 10^5$$
$$1 \leq x \leq k \leq n$$
$$x <= Number\ of\ 'b'\ in$$

Question 3 Max. score: 20.00

1 A Beautiful Sequence!

2 You are given a string $s = s_1 s_2 \cdots s_n$ of length ,

3 which only contains of letters a and b.

Find the **lexicographically smallest** possible **subsequence** of s of **length** k having **atleast** x $'b'$.

String A is lexicographically smaller than string B if it is shorter than B $(|A| < |B|)$ and is its prefix, or if none of them is a prefix of the other and at the first position where they differ character in A is smaller than the character in B.

A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements without changing the order of the remaining elements.

Note

It is guaranteed that answer always exist.