Kartik Bhaiya has a string consisting of only 'a' and 'b' as the characters. Kartik Bhaiya describes perfectness of a string as the maximum length substring of equal characters. Kartik Bhaiya is given a number **k** which denotes the maximum number of characters he can change. Find the maximum perfectness he can generate by changing no more than **k** characters.

Input Format

The first line contains an integer denoting the value of K. The next line contains a string having only 'a' and 'b' as the characters.

Constraints

```
2 ≤ N ≤ 10^6
```

Output Format

A single integer denoting the maximum perfectness achievable.

Sample Input

```
2
abba
```

Sample Output

```
4
```

Explanation

```
We can swap the a's to b using the 2 swaps and obtain the
string "bbbb". This would have all the b's and hence the
answer 4.
Alternatively, we can also swap the b's to make "aaaa". The
final answer remains the same for both cases.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int k = sc.nextInt();
        String s = sc.next();

        int a = perfectness_maximum_length(s,k,'a');
        int b = perfectness_maximum_length(s,k,'b');
        System.out.print(Math.max(a,b));
}

public static int perfectness_maximum_length(String s, int k, char ch){
    int si=0;
    int ei=0;
    int flip = 0;
    int ans=0;
    while(ei < s.length()){
        // Grow window
        if(s.charAt(ei) == ch){
            flip++;
        }

        // shrink window

        while(flip > k){
            if(s.charAt(si)==ch){
                flip--;
            }
            si++;
        }

        // calculate window size
        ans = Math.max(ans, ei - si + 1);
        ei++;
    }
    return ans;
    }

}
```

Given a string, find the first non-repeating character in it. For example, if the input string is “GeeksforGeeks”, then output should be ‘f’ and if input string is “GeeksQuiz”, then output should be ‘G’.

Input Format

The first line contains T denoting the number of testcases. Then follows description of testcases. Each case begins with a single integer N denoting the length of string. The next line contains the string S.

Constraints

Output Format

For each testcase, print the first non repeating character present in string. Print -1 if there is no non repeating character.

Sample Input

```
3
codingblocks
abbac
java
```

Sample Output

```
d
c
j
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc =  new Scanner(System.in);
        int t = sc.nextInt();
        for(int i=0; i<t; i++){
            String s = sc.next();
            if(non_repeating(s) != '0'){
                System.out.println(non_repeating(s));
            }
            else{
                System.out.println(-1);
            }

        }
    }

    public static char non_repeating(String s){
        int [] freq_c = new int [123];
        for(int i=0; i<s.length(); i++){
            char c = s.charAt(i);
            freq_c[c]++;
        }

        for(int i=0; i<s.length(); i++){
            if(freq_c[s.charAt(i)]==1){
                return s.charAt(i);
            }
        }
        return '0';
    }
}
```

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

Input Format

First Line Contains 2 strings of length not more than 10^5

Constraints

```
1<=|S|<=10^5
```

Output Format

A single Line a containing String

Sample Input

```
ADOBECODEBANC ABC
```

Sample Output

```
BANC
```

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        String t = sc.next();
        System.out.print(MinimumWindow(s,t));
    }

            public static String MinimumWindow(String s, String t) {
        int [] frequency_t = new int[123];
        for(int i = 0; i<t.length(); i++) {
            char ch = t.charAt(i);
            frequency_t[ch]++;
        }

        int [] frequency_s = new int[123];
        int si = 0;
        int ei = 0;
        int start = -1;
        int len = Integer.MAX_VALUE;
        int count = 0;
        while(ei < s.length()) {

            char ch = s.charAt(ei);
            frequency_s[ch]++;

            if(frequency_t[ch] >= frequency_s[ch]) {
                count++;
            }

            if(count == t.length()) {
                while(frequency_s[s.charAt(si)] >
frequency_t[s.charAt(si)]) {
                    frequency_s[s.charAt(si)]--;
                    si++;
                }


            if(len > ei - si + 1) {
                len = ei - si + 1;
                start = si;
                }
            }
            ei++;

        }

        if(start == -1) {
            return "";
        }

        return s.substring(start,start+len);
    }

}
```

You have to given a data stream terminated by -1 and the size of sliding window. For each variation in sliding window you need to tell the average of data in current sliding window. Print 4 digits after the decimal point.

Input Format

First line contains an integer denoting the size of sliding window.
Second line has a data stream terminated by -1.

Constraints

```
None
```

Output Format

Print average of each sliding window.

Sample Input

```
5
51 62 24 51 79 33 72 78 84 42 -1
```

Sample Output

```
51.0000 56.5000 45.6667 47.0000 53.4000 49.8000 51.
```

Explanation

```
None
```

```java
import java.util.*;
import java.util.Formatter;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int window_size = sc.nextInt();
        ArrayList<Integer> arr = new ArrayList<>();
        while(true){
            int n = sc.nextInt();
            if(n==-1){
                break;
            }
            arr.add(n);
        }

        int si = 0;
        int ei = 0;
        float Sum = 0;
        int count = 0;
        while(ei < arr.size()){
            Sum += arr.get(ei);
            count++;

            System.out.format("%.4f",Sum/count);
            System.out.print(" ");
            while(count >= window_size){
                Sum -= arr.get(si);
                count--;
                si++;
            }
            ei++;
        }
    }
}
```

Given a string s, find the length of the longest substring without repeating characters.

Input Format

Input string

Constraints

```
0<=|s|<=10000
```

Output Format

Length of longest substring with non repeating characters

Sample Input

```
ABDEFGABEF
```

Sample Output

```
6
```

Explanation

For “ABDEFGABEF”, the longest substring are “BDEFGA” and “DEFGAB”, with length 6.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        int length = s.length();
        int start = 0;
        ArrayList<Character> list = new ArrayList<>();
        int max_length = 0;
        while(start < length){
            while(is_exists(list, s.charAt(start))){
                max_length = Math.max(max_length,list.size());
                list.remove(0);
            }
            list.add(s.charAt(start));
            start++;
        }
        System.out.print( Math.max(max_length,list.size()));
    }

        public static boolean is_exists(ArrayList<Character> list, char
c){
        for(int i=0; i<list.size(); i++){
            if(list.get(i) == c){
                return true;
            }
        }
        return false;
    }
}
```

Given an array of positive numbers, the task is to find the number of possible contiguous subarrays having a product less than a given number k.

Input Format

First line contains Integer where N is the Size of Array
Second line contains Integer k
Next N Line Contains an Integer which denotes element of array

Constraints

```
1<=n<=10^5
1<=k<=10^15
1<=a[i]<=10^5
```

Output Format

Print number of possible contiguous subarrays having product less than a given number k.

Sample Input

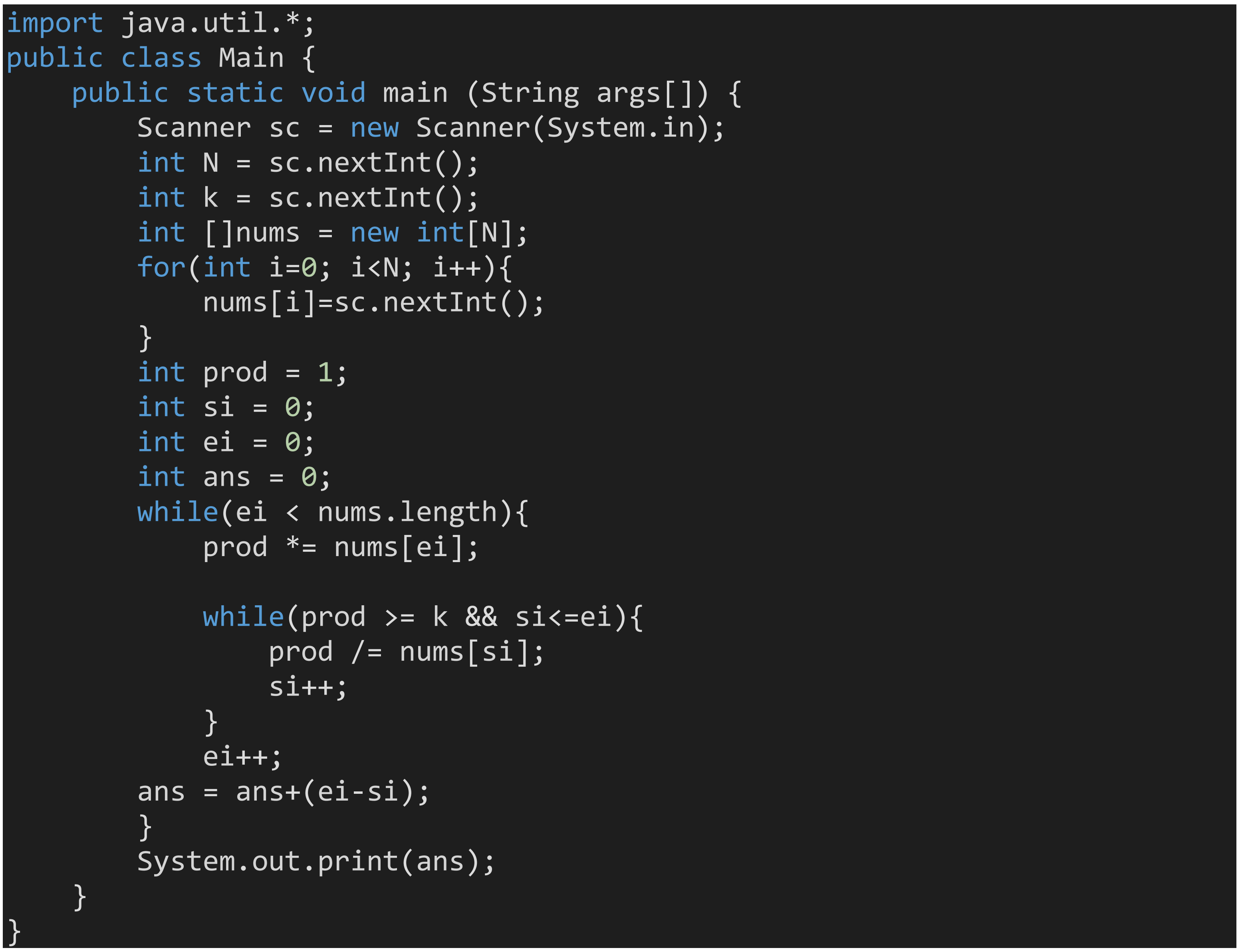
```
4
10
1
2
3
4
```

Sample Output

```
7
```

Explanation

```
The contiguous subarrays are {1}, {2}, {3}, {4} {1, 2}, {1, 2, 3} and
```



```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int k = sc.nextInt();
        int []nums = new int[N];
        for(int i=0; i<N; i++){
            nums[i]=sc.nextInt();
        }
        int prod = 1;
        int si = 0;
        int ei = 0;
        int ans = 0;
        while(ei < nums.length){
            prod *= nums[ei];

            while(prod >= k && si<=ei){
                prod /= nums[si];
                si++;
            }
            ei++;
        ans = ans+(ei-si);
        }
        System.out.print(ans);
    }
}
```