HW_First non-repeating character in a stream 3

Given an input stream A of n characters consisting only of lower case alphabets. While reading characters from the stream, you have to tell which character has appeared only once in the stream upto that point. If there are many characters that have appeared only once, you have to tell which one of them was the first one to appear. If there is no such character then append '#' to the answer. NOTE: 1. You need to find the answer for every i (0 <= i < n) 2. In order to find the solution for every i you need to consider the string from starting position till ith position.

Input Format

The first line will represent String s with n characters

Constraints

1 <= n <= 10^5

Output Format

returns a string after processing the input stream.

Sample Input 0

```
aabc
```

Sample Output 0

```
a#bb
```

Explanation 0

For every ith character we will consider the string from index 0 till index i first non repeating character is as follow- "a" - first non-repeating character is 'a' "aa" - no non-repeating character so '#' "aab" - first non-repeating character is 'b' "aabc" - there are two non repeating characters 'b' and 'c', first non-repeating character is 'b' because 'b' comes before 'c' in the stream.

Submitted Code — with StringBuilder

Language: Java 8

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.next();

        Queue<Character> que = new LinkedList<>();

        int [] frq = new int[26];
        StringBuilder sb = new StringBuilder();

        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
            frq[ch-'a']++;
            que.add(ch);

            while(!que.isEmpty() && frq[que.peek()-'a']>1){
                que.poll();
            }
            if(que.isEmpty()){
                sb.append('#');
            }else{
                sb.append(que.peek());
            }
        }
        System.out.print(sb);
    }
}
```

Submitted Code — without StringBuilder

Language: Java 8

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.next();

        Queue<Character> que = new LinkedList<>();

        int [] frq = new int[26];
        String result ="";

        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
            frq[ch-'a']++;
            que.add(ch);

            while(!que.isEmpty() && frq[que.peek()-'a']>1){
                que.poll();
            }
            if(que.isEmpty()){
                result+='#';
            }else{
                result+=que.peek();
            }
        }
        System.out.print(result);
    }
}
```

i) if a non-repeating

a#bbc

ii) if repeating

firstCharacter as it is

rest #

aaaaa

a####

ii) if non-repeating

abcde

aaaaa

Logic

i) queue

ii) frequency

aabc

for(i - n)

[a]

queue.add(a)

while(freq(que.peek - 'a')>1)

qu.poll()

if(qu.isEmpty)

#

else character

HW_Take Gifts From the Richest Pile 2

You are given an integer array **gifts** denoting the number of gifts in various piles. Every second, you do the following:

Choose the pile with the maximum number of gifts. If there is more than one pile with the maximum number of gifts, choose any. Leave behind the floor of the square root of the number of gifts in the pile. Take the rest of the gifts.

Return the number of **gifts** remaining after **k** seconds.

Input Format

First line contains an integer **n**.

Second line contains an integer array of size **n**.

Third line contains an integer **k**.

Constraints

```
1 <= gifts.length <= 10^3
1 <= gifts[i] <= 10^9
1 <= k <= 10^3
```

Output Format

Returns an integer value.

Sample Input 0

```
5
25 64 9 4 100
4
```

Sample Output 0

```
29
```

Submitted Code

Language: Java 8

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();

        int k= sc.nextInt();

        PriorityQueue<Integer> pq = new PriorityQueue<>((a,b)->b-a);

        int sum =0;
        for(int i=0;i<n;i++){
            sum+=arr[i];
            pq.add(arr[i]);
        }

        for(int i=0;i<k;i++){
            int max = pq.poll();
            int root = (int)Math.sqrt(max);
            sum-=(max-root);
            pq.add(root);
        }
        System.out.print(sum);
    }
}
```

Submitted Code

Language: Java 8

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();

        int k= sc.nextInt();

        PriorityQueue<Integer> pq = new PriorityQueue<>((a,b)->b-a);

        int sum =0;
        for(int i=0;i<n;i++){

            pq.add(arr[i]);
        }

        for(int i=0;i<k;i++){
            int max = pq.poll();
            int root = (int)Math.sqrt(max);
           pq.add(root);
        }

        while(pq.size()>0)sum+=pq.poll();

        System.out.print(sum);
    }
}
```