Longest Substring Without Repeating Characters 6

2 pointer — aquire - release strategy / sliding window.

You are given a **string**, print the length of **Longest Substring** Without Repeating Characters. → unique.

Sample Input 0

```
abcabcbb
```

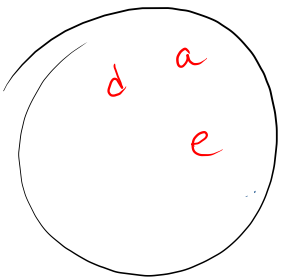
Sample Output 0

```
3
```

ans = 0 1 2 3 4 5

a b a c d c b e a d e

0 1 2 3 4 5 6 7 8 9 10



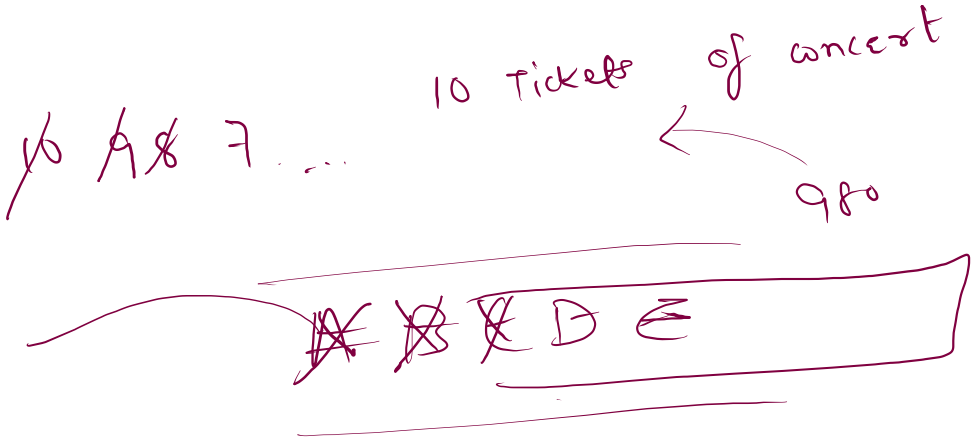

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        int i = 0;
        int j = 0;
        int ans = 0;
        HashSet<Character> hs = new HashSet<>();
        while(j < s.length()){
            if(hs.contains(s.charAt(j))){  //release
                hs.remove(s.charAt(i));
                i++;
            }else{                    //acquire
                hs.add(s.charAt(j));
                j++;
            }
            ans = Math.max(ans, hs.size());
        }
        System.out.println(ans);
    }
}
```

j

Queue → pipe like D.S (FIFO) ⇒ First in first out



Queue.

⇓

Interface.

- initialize
- add
- remove
- get
- size

Stack = new Stack()

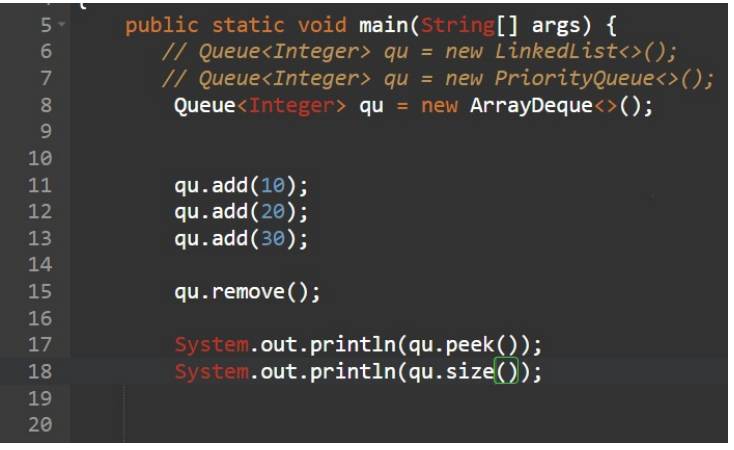
AL = new ArrayList

HashMap = new Hashmap

Hashset = new HashSet

obj

Queue = ~~new Queue~~



```
public static void main(String[] args) {
   // Queue<Integer> qu = new LinkedList<>();
   // Queue<Integer> qu = new PriorityQueue<>();
    Queue<Integer> qu = new ArrayDeque<>();


    qu.add(10);
    qu.add(20);
    qu.add(30);

    qu.remove();

    System.out.println(qu.peek());
    System.out.println(qu.size());
```

Queue Syntax Learning

1. **Declare an Empty** $queue\ s$.
2. Take Single Integer T as input.
3. For next T Lines format $(case, x(optional))$

- case 1. $Print$ the $size$ of the $queue$ in a separate line.
- case 2. $Remove$ an element from the queue. If the queue is empty then print -1 in a separate line.
- case 3. Add Integer x to the $queue\ s$.
- case 4. $Print$ an element at the $front$ of the $queue$. If queue is empty print -1 in a seperate line.

Sample Input 0

```
5
1
2
3 9
4
1
```

Sample Output 0

```
0
-1
9
1
```

```java
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int t = scn.nextInt();
        Queue<Integer> qu = new LinkedList<>();

        for(int i = 0; i < t; i++){
            int cNu = scn.nextInt();
            if(cNu == 1){
                System.out.println(qu.size());
            }else if(cNu == 2){
                if(qu.size() == 0){
                    System.out.println(-1);
                }else{
                    qu.remove();
                }
            }else if(cNu == 3){
                int x = scn.nextInt();
                qu.add(x);
            }else{
                if(qu.size() == 0){
                    System.out.println(-1);
                }else{
                    System.out.println(qu.peek());
                }
            }
        }
    }
}
```

binary ⇒ 12

$(12)_{10} \rightarrow (\quad)_2$

$(15)_{10} \rightarrow (\quad)_2$

$(14)_{10} \rightarrow (\quad)$

Print Binary

algo { remove print add 2 more } n times.

Given a number **N**. The task is to generate and print all **binary numbers** with decimal values from **1 to N.** (Note : Use the **queue** for implementation.)

1 . 1
10 2
11 3
100 4
101 5
110 6
111 7
1000 8
1001 9

Sample Input 0

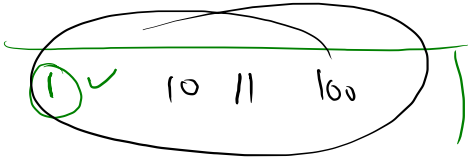
```
4
```

Sample Output 0

```
1 10 11 100
```

Explanation 0

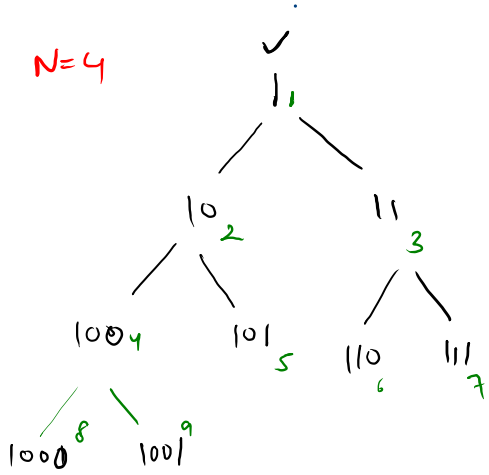
- 1(Decimal) ⇒ 1(Binary)
- 2(Decimal) ⇒ 10(Binary)
- 3(Decimal) ⇒ 11(Binary)
- 4(Decimal) ⇒ 100(Binary)



101 110 111 1000 1001

String

N=4



N=5

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        Queue<String> qu = new LinkedList<>();
        qu.add("1");

        //algo -- remove / print / add 2 more } n times
        for(int i = 0; i < n; i++){
            String rem = qu.remove();
            System.out.print(rem + " ");
            qu.add(rem + "0");
            qu.add(rem + "1");
        }

    }
}
```

First Negative Integer 2

Given an array **A[]** of size **N** and a positive integer **K**, find the first negative integer for each and every window(**contiguous subarray**) of size **K**.

Sample Input 0

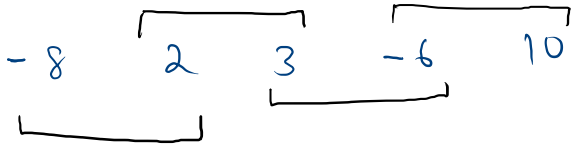
```
5 2
-8 2 3 -6 10
```

Sample Output 0

```
-8 0 -6 -6
```

N = 5

k = 2



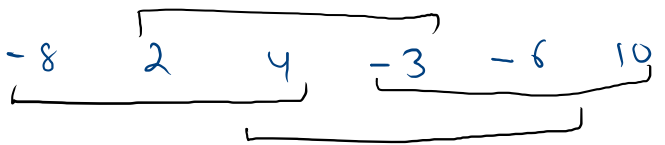
N = 6

k = 2

-8 0 -3 -3 -6

N = 6

k = 3



-8 -3 -3 -3

k=3 N=9

-8	2	-2	-3	4	5	6	-7	8
0	1	2	3	4	5	6	7	8

i

i

n-1 ≃ n

n-k

dig = -8

indexOf

TC = ?

x

for 1st k ele.

1. add -ve.

rest

2. find one prev

3. remove unnecessary

A[i] < 0

A[i]

0

peek < i-k+1

4. add -ve

DS SC↑ TC↓

O(nx)