ASSIGNMENT 2

Question 1: Our institute has arranged a blood donation camp. The students are advised to donate blood, and you as the head boy/girl have to maintain the records. There are two lists A and B. List A has the order of the roll number of students who have donated blood. List B is initially empty.

A student can donate blood more than one time. Each time a student donates blood you have to find the roll number of student who has donated blood only once and was earliest in list A and append the roll number in list B.

You are given list A as Input and you have to print list B

INPUT

First-line will be the number of students, "n" (0<n<100000)
Second-line will be the size of list A, "s" (0<s< 100000)
The next line will contain "s" integers which are the roll number of students.

"Roll number will be from 1 to n "

OUTPUT

You have to print list B which will be of size s.

Note: If there is no such student then append 0.

Test Case 1:

Input

```
7
10
2 1 3 5 2 4 6 3 1 7
```

Output

```
2 2 2 2 1 1 1 1 5 5
```

Explanation: Here 7 is the number of students and 10 is the size of list A.
Step 1 - 2 has donated blood at first and is the only person to donate 1 time.
Step 2 - 1 has donated blood. Both 1,2 have donated one time each, but since 2 comes before in list A, 2 is printed.
Step 3, Step 4 - 2 is printed as in the previous steps.
Step 5 - As now 2 has donated twice, the next student is who has donated once is 1, so 1 is printed…

Step 9 - 2, 1, 3 have donated blood two times so 5 is appended.
Test Case 1:

```
6
5
1 2 1 2 3 4
```

Output

```
1 1 2 0 3 3
```

NOTE: IMPLEMENT THE PROBLEM USING QUEUE AND ARRAY. TIME AND SPACE COMPLEXITY SHOULD BE O(N).

Question 2-

You are given an initial stack S1 of size N represented as :
For example a stack of size 5 -
[5 4 3 2 1]
where 5 is at the bottom and 1 is at the top.

Now you are given two more stacks S2 and S3. Now, using these two stacks you have to put all the numbers in stack S2 or S3 in a given order O which is a permutation of given numbers. Given , you have to answer if the order O is possible by transferring the numbers from stack S1to stack S2 using operations of stack.

Input:
N : size of stack S1
S1 : initial stack
O : final order in stack S2

Output:
Yes: if the order O is possible
No : if the order O is not possible

Sample Case 1:
S1 : [5 4 3 2 1]
O : [1 2 3 4 5]

Output : Yes

Explanation:

Iteration #1:
S1: [5 4 3 2 1]
S2: []
S3: []

Iteration #2:
S1: [5 4 3 2]
S2: [1]
S3: []

Iteration #3:
S1: [5 4 3]
S2: [1 2]
S3: []

Iteration #4:
S1: [5 4]
S2: [1 2 3]
S3: []

Iteration #5:
S1: [5]
S2: [1 2 3 4]
S3: []

Iteration #6:
S1: []
S2: [1 2 3 4 5]
S3: []

Sample Case 2:
S1 : [5 4 3 2 1]
O : [1 2 3 5 4]

Output : Yes

Sample Case 3:
S1 : [5 4 3 2 1]
O : [3 1 4 2 5]

Output : No

Question 3:
You are given a stack S1 of size N as such:
S= [5 4 3 2 1]
You are given another stack S2. You have three operations in stack S1: push, pop and delay.

Push puts the element at the top of the stack.
Pop deletes the top element of the stack.
Delay just neglects the number and considers the next element from the top.

The delayed elements are popped in the order such that the element delayed first is popped first.

The elements popped from stack S1 or from the delayed elements are pushed into stack S2.

You are given an order O of stack S2. You have to answer if the given order O is possible in stack S2 using these given operations.
You are free to use additional stack/queue.

Input -
N : size of stack S1
S1 : initial stack
O : order of stack S2

Output:
Yes: if the order O is possible
No : if the order O is not possible

Sample Case 1:
S1 = [5 4 3 2 1]
O= [1 3 5 2 4]

Output:
Yes

Explanation:
S1 = [5 4 3 2 1]
S2= []
Delayed= []

S1 = [5 4 3 2]
S2= [1]
Delayed= []

S1 = [5 4 3]
S2= [1]
Delayed= [2]

S1 = [5 4]
S2= [1 3]
Delayed= [2]

S1 = [5]
S2= [1 3]
Delayed= [2 4]

S1 = []
S2= [1 3 5]
Delayed= [2 4]

S1 = []
S2= [1 3 5 2]
Delayed= [4]

S1 = []
S2= [1 3 5 2 4]
Delayed= []

Sample Case 2:
S1 = [5 4 3 2 1]
O= [1 2 5 4 3]

Output:
No

Question 4: You are given an array A. for each element in array A you have to find the next greatest element on the right side of the array. If there is no greater element take -1 as the greater element.

Input format:

First-line will be the size of the array, “n” (0<n<100000)
The next line will contain “n” integers.

Output:

Print n integers .

Test Case 1:

Input

```
10
2 1 3 5 2 4 6 3 1 7
```

Output

```
3 3 5 6 4 6 7 7 7 -1
```

NOTE: IMPLEMENT THE PROBLEM USING STACK AND ARRAY. TIME AND SPACE COMPLEXITY SHOULD BE O(N).

Q1> Algo:

```
1. Take input from user
      N ← no. of students
      a-size ← count of students donated blood
      a ← sequence in which student donated blood

2. Initialize
      counter ← no. of times student donated blood
      b ← unique list representing students

3. Iterate for each student in 'a'
   3.1 Increment counter for that student
       representing number of times student
       donated blood.
   3.2 push to 'b' student name if student
       donating blood first time

   3.3 If student has given blood only once
       print front of queue
       else
           Iterate 'b' and find the student
           who has given blood only once

       If 'b' is empty
           print 0
       else
           print queue front
```

How to execute

Run the code in any Java compiler

Provide inputs in sequence as given in program

Q2) Algo:

```
1.  Initialize stack s1, s2, s3
                array output representing O

2.  int output_ptr = 0  ← represents the element
                          of the output list

3.  for each element in s1

        int curr = s1.pop()

        If (curr == output_ptr())
            push curr to s2
            output_ptr++
        else
            push curr to s3

4.  Merge s3 to s2

5.  Compare s2 and output

        If same print "Yes"
        else    print "No"
```

How to execute

Run the code in any Java compiler

Provide inputs in sequence as given in program

Q3) <u>Algo:</u>

```
1.  Initialize stack s1, s2,
            array output representing O
            array delayed

2.  int output_ptr = 0  ← represents the element
                          of the output list

3.  for each element in s1

        int curr = s1.pop()

        If (curr == output_ptr())
            push curr to s2
            output_ptr++
        else
            add curr to delayed

4.  Merge delayed to s2

5.  Compare s2 and output

        If same print "Yes"
        else print "No"
```

How to execute

Run the code in any Java compiler

Provide inputs in sequence as given in program