Q1. You are given a string of brackets i.e. '{', '}' , '(' , ')', '[' , ']' . You have to check whether the sequence of parenthesis is balanced or not.

For example, "(())", "(())()" are balanced and "())(", "(()))" are not.

Input Format

A string of '(' , ')' , '{' , '}' and '[' , ']' .

Constraints

1<=|S|<=10^5

Output Format

Print "Yes" if the brackets are balanced and "No" if not balanced.

Sample Input

```
(())
```

Sample Output

```
Yes
```

Explanation

(()) is a balanced string.

SOLUTION:

```
import java.util.*;

public class Main {
    static boolean balancedParenthesis(String str) {
        Stack<Character> s = new Stack<>();
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            if (c == '(' || c == '{' || c == '[') {
                s.push(c);
            } else if (c == ')') {
                if (s.empty() || s.peek() != '(') {
                    return false;
                }
                s.pop();
            } else if (c == '}') {
                if (s.empty() || s.peek() != '{') {
                    return false;
```

```
                }
                s.pop();
            } else if (c == ']') {
                if (s.empty() || s.peek() != '[') {
                    return false;
                }
                s.pop();
            }
        }
        return s.empty();
    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        String str = sc.next();
        boolean ans = balancedParenthesis(str);
        if (ans)
            System.out.println("Yes");
        else
            System.out.println("No");

    }
}
```

Q2. John and Mike are both trying to book few tickets available for a movie show. Write a Program using Multithreading to book tickets ensuring required tickets can only be booked if available tickets are more for each of them. **(Creating, Evaluating)**

Shown below is the class having main method. Complete rest of the code.

```
import java.util.Scanner;
public class TicketBooking {
   public static void main(String[] args) {
      Scanner tk = new Scanner(System.in);
      int availTickets = tk.nextInt();
      int reqJohn = tk.nextInt();
      int reqMike = tk.nextInt();
      AvailableTicket avlTic = new AvailableTicket(availTickets,reqJohn,reqMike);
      Thread t = new Thread(avlTic);
      Thread tt = new Thread(avlTic);
      t.setName("John");
```

```
        tt.setName("Mike");
        t.setPriority(10);
        t.start();
        tt.start();
    }
}
```

Input format:

First line of the input contains available tickets. Second line contains tickets required by John and third line contains tickets required by Mike.

Output format:

Each line of the output contains *currentThread* name (either John or Mike) followed by either ticket booked or not. If ticket is booked then after *currentThread* print ": tickets booked: " and if available tickets is less than required by either of them then print ": not booked.". Print the exact message considering the spaces.

Sample Input:

5

3

2

Sample Output:

John: tickets booked: 3

Mike: tickets booked: 2

Sample Input:

6

4

3

Sample Output:

John: tickets booked: 4

Mike: not booked

Q3 https://practice.geeksforgeeks.org/problems/number-following-a-pattern3126/1

Q4. Kartik Bhaiya, mentor at Coding Blocks, organized a party for their interns at Coding Blocks. In a party of N people, only one person is known to everyone. Such a person may be present in the party, if yes, she/he doesn't know anyone in the party. We can only ask questions like "does A know B? ".

Find the stranger (celebrity) in minimum number of questions.

Input Format

First line contains **N**, number of persons in party. Next line contains the matrix of N x N which represents A knows B when it's value is one.

Constraints

None

Output Format

Print the celebrity ID which is between 0 and N-1. If celebrity is not present then print "No Celebrity".

Sample Input

```
4
0 0 1 0
0 0 1 0
0 0 0 0
0 0 1 0
```

Sample Output

```
2
```

Explanation

In the given case there are 4 persons in the party let them as A, B, C, D. The matrix represents A knows B when it's value is one. From the matrix, A knows C, B knows C and D knows C. Thus C is the celebrity who doesnot know anyone and it's ID is 2. Hence output is 2.

SOLUTION:

```
import java.util.Stack;

class Main{
    // Person with 2 is celebrity
    static int MATRIX[][] = { { 0, 0, 1, 0 },
```

```
                            { 0, 0, 1, 0 },
                            { 0, 0, 0, 0 },
                            { 0, 0, 1, 0 } };

// Returns true if a knows
// b, false otherwise
static boolean knows(int a, int b)
{
    boolean res = (MATRIX[a][b] == 1) ? true : false;
    return res;
}

// Returns -1 if celebrity
// is not present. If present,
// returns id (value from 0 to n-1).
static int findCelebrity(int n)
{
    Stack<Integer> st = new Stack<>();
    int c;

    // Step 1 :Push everybody
    // onto stack
    for (int i = 0; i < n; i++) {
        st.push(i);
    }

    while (st.size() > 1) {
        // Step 2 :Pop off top
        // two persons from the
        // stack, discard one
        // person based on return
        // status of knows(A, B).
        int a = st.pop();
        int b = st.pop();

        // Step 3 : Push the
        // remained person onto stack.
        if (knows(a, b)) {
            st.push(b);
        }

        else
            st.push(a);
    }

    // If there are only two people
    // and there is no
    // potential candidate
    if (st.empty())
        return -1;

    c = st.pop();

    // Step 5 : Check if the last
    // person is celebrity or not
    for (int i = 0; i < n; i++) {
```

```
            // If any person doesn't
            //  know 'c' or 'a' doesn't
            // know any person, return -1
            if (i != c && (knows(c, i) || !knows(i, c)))
                return -1;
        }
        return c;
    }

    // Driver Code
    public static void main(String[] args)
    {
        int n = 4;
        int result = findCelebrity(n);
        if (result == -1) {
            System.out.println("No Celebrity");
        }
        else
            System.out.println("Celebrity ID " + result);
    }
}
```

Q5. We are given a circular array, print the next greater number for every element. If it is not found print -1 for that number. To find the next greater number for element A_i , start from index i + 1 and go uptil the last index after which we start looking for the greater number from the starting index of the array since array is circular.

Input Format

First line contains the length of the array n. Second line contains the n space separated integers.

Constraints

1 <= n <= 10^6
-10^8 <= A_i <= 10^8 , 0<= i< n

Output Format

Print n space separated integers each representing the next greater element.

Sample Input

```
3
1 2 3
```

Sample Output

```
2 3 -1
```

Explanation

Next greater element for 1 is 2,
for 2 is 3 but not present for 3 therefore -1

SOLUTION:

```
public static int[] nextGreaterElements(int[] nums) {
    int[] res = new int[nums.length];

        Stack stack = new Stack<>();

        for (int i = 2 * nums.length - 1; i >= 0; i--) {

            while (!stack.isEmpty() && nums[stack.peek()] <= nums[i %
nums.length]) {
                stack.pop();
            }

            res[i % nums.length] = (stack.isEmpty() ? -1 :
nums[stack.peek()]);
            stack.push(i % nums.length);
        }

        return res;

    }
```

Q6. Given a linked list with n nodes. Find the k^{th} element from last without computing the length of the linked list.

Input Format

First line contains space separated integers representing the node values of the linked list. The list ends when the input comes as '-1'. The next line contains a single integer k.

Constraints

n < 10^5

Output Format

Output a single line containing the node value at the kth element from last.

Sample Input

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

Sample Output

```
4
```

Explanation

The linked list is 1 2 3 4 5 6. -1 is not included in the list. So the third element from the last is 4

Q7

https://practice.geeksforgeeks.org/problems/stock-span-problem-1587115621/1

Q8

https://leetcode.com/problems/largest-rectangle-in-histogram/

Q9 The Game is as follows You have given a binary array, where 1 denotes push operation and 0 denotes a pop operation in a queue. The task is to check if the possible set of operations are valid or not.
Print Valid if the set of Operations are Valid Otherwise Print Invalid.

Input Format

The First Line contains an Integer T, as the number of Test cases.
The Next Line contains an Integer N, as the Size of the Array.
The Next Line contains N Binary numbers separated by space.

Output Format

Print *Valid* If the set of operations are valid Otherwise Print *Invalid* for each Test Case separated by a new Line.

Sample Input

```
2
5
1 1 0 0 1
5
1 1 0 0 0
```

Sample Output

```
Valid
Invalid
```

SOLUTION:

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

            int t = sc.nextInt();
            while(t-->0) {
            int n = sc.nextInt();
            int arr[] = new int[n];
            for (int i = 0; i < arr.length; i++) {
                  arr[i] = sc.nextInt();
            }
            System.out.println(Queue(arr));
```

```
        }
    }
    public static String Queue(int [] arr) {
        Stack<Integer>s= new Stack<>();

        for (int i = 0; i < arr.length; i++) {
            if(arr[i]==1) {
                s.push(arr[i]);

            }
            if(arr[i]==0) {
                if(s.isEmpty()) {
                    return "Invalid";
                }
                s.pop();
            }


        }
        return "Valid";

    }

}
```

Q10
https://practice.geeksforgeeks.org/problems/next-larger-element-1587115620/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article

Q11. Given an array **arr[]** of **n** integers. Check whether it contains a triplet that sums up to zero.

Input Format

First line contain size of the array.

Next line is the n integers of the array.

Output Format

Print 1 if the triplet with 0 exists and 0 if it doesn't exists.

Sample Input
5
0 -1 2 -3 1

Sample Output
1

Explanation
0, -1 and 1 forms a triplet with sum equal to 0.

SOLUTION:

```
import java.util.*;
public class Main
{
  static public boolean findTriplets(int arr[] , int n)
```

```
    {
        //add code here.
        boolean found=false;
        for(int i=0;i<n-1;i++){
            HashSet<Integer> hs=new HashSet<>();
            for(int j=i+1;j<n;j++){
                int x=-(arr[i]+arr[j]);
                if(hs.contains(x)){
                    return true;
                }else{
                    hs.add(arr[j]);
                }
            }
        }
        return false;
    }

    public static void main(String args[]) {
        int n;
        Scanner sc=new Scanner(System.in);
        n=sc.nextInt();

        int [] arr=new int[n];

        for(int i=0;i<n;i++)
        arr[i]=sc.nextInt();

        if(findTriplets(arr,n))
        System.out.println("1");

        else
         System.out.println("0");


    }
}
```

Q12.Given an integer array print 1 if any value appears **at least twice** in the array, and print 0 if every element is distinct.

Input format:

First line contain size of the array.

Next line is the n integers of the array.

Output Format

Print 1 if any element occur at least twice and 0 if every element is distinct.

SOLUTION:

```
import java.util.*;
public class Main {

   public static boolean containsDuplicate(int[] nums)
{
      HashMap<Integer,Integer> map = new HashMap<>();
      for (int i = 0; i < nums.length; i++) {
         if (map.containsKey(nums[i])) {
            return true;
         }
         map.put(nums[i],1);
      }
      return false;
  }

   public static void main(String args[]) {
      int n;
      Scanner sc=new Scanner(System.in);
      n=sc.nextInt();

      int [] arr=new int [n];

      for(int i=0;i<n;i++)
      arr[i]=sc.nextInt();

      if(containsDuplicate(arr))
      System.out.print("1");

      else
      System.out.print("0");



   } }
```