

Q.6

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

    /\*

     \* Complete the 'doStringReverse' function below.

     \*

     \* The function is expected to return a STRING.

     \* The function accepts STRING value as parameter.

     \*/

    public static String doStringReverse(String value) {

         int len=value.length();

         int i;

        String rev="";

        for( i=len-1; i>=0;i--)

        {

            // System.out.println(value);

           rev=rev+value.charAt(i);

        }

        return rev;

    }

}

public class Solution {

    public static void main(String[] args) throws IOException {

        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

        String value = bufferedReader.readLine();

        String result = Result.doStringReverse(value);

        bufferedWriter.write(result);

        bufferedWriter.newLine();

        bufferedReader.close();

        bufferedWriter.close();

    }

}

Q.7

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

// Java implementation of simple method to find count of

// pairs with given sum.

public class Solution

{

    public static void main(String args[])

    {

        // int[] arr = { 1, 5, 7, -1, 5 };

        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 sum = sc.nextInt();

        getPairsCount(arr, sum);

    }

    // Prints number of pairs in arr[0..n-1] with sum equal

    // to 'sum'

    public static void getPairsCount(int[] arr, int sum)

    {

        int count = 0;// Initialize result

        // Consider all possible pairs and check their sums

        for (int i = 0; i < arr.length; i++)

            for (int j = i + 1; j < arr.length; j++)

                if ((arr[i] + arr[j]) == sum)

                    count++;

        System.out.print(count);

    }

}

Q.8

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    static void Max\_Sum(int arr[], int n, int k)

{

    // To store the break point

    int p = n;

    // Sort the given array

    Arrays.sort(arr);

    // Find the break point

    for (int i = 0; i < n; i++)

    {

        // No need to look beyond i'th index

        if (arr[i] >= k)

        {

            p = i;

            break;

        }

    }

    int maxsum = 0, a = 0, b = 0;

    // Find the required pair

    for (int i = 0; i < p; i++)

    {

        for (int j = i + 1; j < p; j++)

        {

            if (arr[i] + arr[j] < k &&

                arr[i] + arr[j] > maxsum)

            {

                maxsum = arr[i] + arr[j];

                a = arr[i];

                b = arr[j];

            }

        }

    }

    // Print the required answer

    System.out.print( a + " " + b);

}

// Driver code

public static void main (String[] args)

{

    // int []arr = {5, 20, 110, 100, 10};

    Scanner sc = new Scanner(System.in);

   int len= sc.nextInt();

    int arr[] = new int[len];

    for (int i = 0; i < len; i++) {

            arr[i] = sc.nextInt();

       }

    int k = sc.nextInt();

    int n = arr.length;

    // Function call

    Max\_Sum(arr, n, k);

}

}

Q.9

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    // static int arr[] = new int[]{10, 12, 13, 16, 18, 19, 20, 21, 22, 23,

    //                                      24, 33, 35, 42, 47};

    // If x is present in arr[0..n-1], then returns

    // index of it, else returns -1.

       public static int interpolationSearch(int[] sortedArray, int toFind)

    {

        int low = 0;

        int high = sortedArray.length - 1;

        int mid;

        while (sortedArray[low] <= toFind && sortedArray[high] >= toFind)

        {

            if (sortedArray[high] - sortedArray[low] == 0)

                return (low + high)/2;

            /\*\* out of range is possible  here \*\*/

             mid = low + ((toFind - sortedArray[low]) \* (high - low)) / (sortedArray[high] - sortedArray[low]);

             if (sortedArray[mid] < toFind)

                 low = mid + 1;

             else if (sortedArray[mid] > toFind)

                 high = mid - 1;

             else

                 return mid;

        }

        if (sortedArray[low] == toFind)

            return low;

           /\*\* not found \*\*/

        else

            return -1;

    }

    /\*\* Main method \*\*/

    public static void main(String[] args)

    {

        Scanner scan = new Scanner( System.in );

        // System.out.println("Interpolation Search Test\n");

        int n, i;

        /\*\* Accept number of elements \*\*/

        // System.out.println("Enter number of integer elements");

        n = scan.nextInt();

        /\*\* Create integer array on n elements \*\*/

        int arr[] = new int[ n ];

        /\*\* Accept elements \*\*/

        // System.out.println("\nEnter "+ n +" sorted integer elements");

        for (i = 0; i < n; i++)

            arr[i] = scan.nextInt();

        // System.out.println("\nEnter element to search for : ");

        int key = scan.nextInt();

        int result = interpolationSearch(arr, key);

        if (result == 1)

        System.out.println();

        else

         System.out.println( result);

    }

}

Q.10

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    public static int jumpSearch(int[] arr, int x)

    {

        int n = arr.length;

        // Finding block size to be jumped

        int step = (int)Math.floor(Math.sqrt(n));

        // Finding the block where element is

        // present (if it is present)

        int prev = 0;

        while (arr[Math.min(step, n)-1] < x)

        {

            prev = step;

            step += (int)Math.floor(Math.sqrt(n));

            if (prev >= n)

                return -1;

        }

        // Doing a linear search for x in block

        // beginning with prev.

        while (arr[prev] < x)

        {

            prev++;

            // If we reached next block or end of

            // array, element is not present.

            if (prev == Math.min(step, n))

                return -1;

        }

        // If element is found

        if (arr[prev] == x)

            return prev;

        return 1;

    }

    // Driver program to test function

    public static void main(String [ ] args)

    {

        // int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21,

        //             34, 55, 89, 144, 233, 377, 610};

        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 x = sc.nextInt();

        // Find the index of 'x' using Jump Search

        int index = jumpSearch(arr, x);

        // Print the index where 'x' is located

        System.out.println(index);

    }

}

Q.11

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

      public static boolean isDivisible(int num, int x, int y)

    {

        // While num divisible is divible

        // by either x or y, keep dividing

        while (num % x == 0 || num % y == 0)

        {

            if (num % x == 0)

                num /= x;

            if (num % y == 0)

                num /= y;

        }

        // If num > 1, it means it cannot be

        // further divided by either x or y

        if (num > 1)

            return false;

        return true;

    }

    // Funcion to calculate gcd of two numbers

    // using Euclid's algorithm

    public static int \_gcd(int a, int b)

    {

        while (a != b)

        {

            if (a > b)

                a = a - b;

            else

                b = b - a;

        }

        return a;

    }

    // Function that returns true if all

    // the array elements can be made

    // equal with the given operation

    public static boolean isPossible(int[] arr, int n,

                                        int x, int y)

    {

        // To store the gcd of the array elements

        int gcd = arr[0];

        for (int i = 1; i < n; i++)

            gcd = \_gcd(gcd, arr[i]);

        // For every element of the array

        for (int i = 0; i < n; i++)

        {

            // Check if k is of the form x\*x\*..\*y\*y\*...

            // where (gcd \* k = arr[i])

            if (!isDivisible(arr[i] / gcd, x, y))

                return false;

        }

        return true;

    }

    // Driver code

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        int len= sc.nextInt();

        int arr[] = new int[len];

        for(int i=0;i<len;i++)

        {

            arr[i]=sc.nextInt();

        }

        // int[] arr = { 2, 4, 6, 8 };

        int n = arr.length;

        int x = sc.nextInt();

        int y = sc.nextInt();

        boolean b=true;

        if (isPossible(arr, n, x, y))

        {

            b=true;

            System.out.println(b);

        }

        else

        {

            b=false;

            System.out.println(b);

        }

    }

}

Q.12

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

// Java implementation of the approach

class Solution {

    // Function to return the largest element

    // such that its previous and next

    // element product is maximum

    static int maxElement(int a[], int n)

    {

        if (n < 3)

            return -1;

        int maxElement = a[0];

        int maxProd = a[n - 1] \* a[1];

        for (int i = 1; i < n; i++) {

            // Calculate the product of the previous

            // and the next element for

            // the current element

            int currProd = a[i - 1] \* a[(i + 1) % n];

            // Update the maximum product

            if (currProd > maxProd) {

                maxProd = currProd;

                maxElement = a[i];

            }

            // If current product is equal to the

            // current maximum product then

            // choose the maximum element

            else if (currProd == maxProd) {

                maxElement = Math.max(maxElement, a[i]);

            }

        }

        return maxElement;

    }

    // Driver code

    public static void main(String[] args)

    {

        // int[] a = { 5, 6, 4, 3, 2 };

        Scanner sc = new Scanner(System.in);

        // System.out.println("Enter Length Of Array:");

        int n=sc.nextInt();

        int a[] = new int[n];

        // System.out.println("Enter Array");

        for (int i=0; i<n; i++)

        {

            a[i] = sc.nextInt();

        }

        System.out.println(maxElement(a, n));

    }

}

// public class Solution {

//     public static void main(String args[] ) throws Exception {

//         /\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

//         int n,max;

//         // BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

//         Scanner sc = new Scanner(System.in);

//         System.out.println("Enter Length Of Array:");

//         n=sc.nextInt();

//         int a[] = new int[n];

//         System.out.println("Enter Array");

//         for (int i=0; i<n; i++)

//         {

//             a[i] = sc.nextInt();

//         }

//         max = a[0];

//         for(int i=0; i<n; i++)

//         {

//             if(a[i]>max)

//             {

//                 max = a[i];

//             }

//             System.out.println(max);

//         }

//     }

// }

Q.13

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    static int Max\_Sum(int arr1[],

                   int arr2[], int n)

{

    // To store dp value

    int [][]dp = new int[n][2];

    // For loop to calculate the value of dp

    for (int i = 0; i < n; i++)

    {

        if(i == 0)

        {

            dp[i][0] = arr1[i];

            dp[i][1] = arr2[i];

            continue;

        }

        dp[i][0] = Math.max(dp[i - 1][0],

                            dp[i - 1][1] + arr1[i]);

        dp[i][1] = Math.max(dp[i - 1][1],

                            dp[i - 1][0] + arr2[i]);

    }

    // Return the required answer

    return Math.max(dp[n - 1][0],

                    dp[n - 1][1]);

}

// Driver code

public static void main(String[] args)

{

    Scanner sc = new Scanner(System.in);

    int len=sc.nextInt();

    int arr1[] = new int[len];

    int arr2[] = new int[len];

    for (int i=0;i<len;i++)

    {

        arr1[i]=sc.nextInt();

    }

     for (int i=0;i<len;i++)

    {

        arr2[i]=sc.nextInt();

    }

    // int arr1[] = {9, 3, 5, 7, 3};

    // int arr2[] = {5, 8, 1, 4, 5};

    int n = arr1.length;

    // Function call

    System.out.println(Max\_Sum(arr1, arr2, n));

}

}

Q.14

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

import java.util.Arrays;

public class Solution {

 public static void main(String[] args) {

        Scanner sc= new Scanner(System.in);

        int len = 10;

        int numbers[] = new int[len];

        for(int i=0;i<len;i++)

        {

            numbers[i]=sc.nextInt();

        }

        int N = sc.nextInt();

        int idealSum = (N \* (N + 1)) / 2;

        int sum = Arrays.stream(numbers).sum();

        int missingNumber = idealSum - sum;

        System.out.println(missingNumber);

    }

}

Q.15

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    static void calculateSpan(int price[], int n, int S[])

    {

        // Span value of first day is always 1

        S[0] = 1;

        // Calculate span value of remaining days by linearly checking

        // previous days

        for (int i = 1; i < n; i++) {

            S[i] = 1; // Initialize span value

            // Traverse left while the next element on left is smaller

            // than price[i]

            for (int j = i - 1; (j >= 0) && (price[i] >= price[j]); j--)

                S[i]++;

        }

    }

    // A utility function to print elements of array

    static void printArray(int arr[])

    {

        for(int i=0; i<arr.length;i++)

        {

        System.out.print(arr[i]+" ");

        }

    }

    // Driver program to test above functions

    public static void main(String[] args)

    {

        // int price[] = { 10, 4, 5, 90, 120, 80 };

        Scanner sc = new Scanner(System.in);

        int len = sc.nextInt();

        int price[] = new int[len];

        for(int i=0;i<len;i++)

        {

            price[i]=sc.nextInt();

        }

        int n = price.length;

        int S[] = new int[n];

        // Fill the span values in array S[]

        calculateSpan(price, n, S);

        // print the calculated span values

        printArray(S);

    }

}

Q.16

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

   static Stack<Integer> sortStack(Stack<Integer> input)

    {

        Stack<Integer> tmpStack =

                       new Stack<Integer>();

        while (!input.empty())

        {

            // pop out the

            // first element

            int tmp = input.peek();

            input.pop();

            // while temporary stack is

            // not empty and top of stack

            // is smaller than temp

            while (!tmpStack.empty() &&

                    tmpStack.peek() < tmp)

            {

                // pop from temporary

                // stack and push it

                // to the input stack

                input.push(tmpStack.peek());

                tmpStack.pop();

            }

            // push temp in

            // tempory of stack

            tmpStack.push(tmp);

        }

        return tmpStack;

    }

    static void sortArrayUsingStacks(int []arr,

                                     int n)

    {

        // push array elements

        // to stack

        Stack<Integer> input =

                       new Stack<Integer>();

        for (int i = 0; i < n; i++)

            input.push(arr[i]);

        // Sort the temporary stack

        Stack<Integer> tmpStack =

                       sortStack(input);

        // Put stack elements

        // in arrp[]

        for (int i = 0; i < n; i++)

        {

            arr[i] = tmpStack.peek();

            tmpStack.pop();

        }

    }

    // Driver Code

    public static void main(String args[])

    {

        // int []arr = {10, 5, 15, 45};

        Scanner sc = new Scanner(System.in);

        int len=sc.nextInt();

        int arr[] =new int[len];

        for(int i=0;i<len;i++)

        {

            arr[i]=sc.nextInt();

        }

        int n = arr.length;

        sortArrayUsingStacks(arr, n);

        for (int i = 0; i < n; i++)

            System.out.print(arr[i] + " ");

    }

}

Q.19

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    static boolean checkPerfectcube(int n)

{

    // Takes the sqrt of the number

    int d = (int)Math.cbrt(n);

    // Checks if it is a perfect

    // cube number

    if (d \* d \* d == n)

        return true;

    return false;

}

// Function to return the smallest perfect

// cube from the array

static int smallestPerfectCube(int a[], int n)

{

    // Stores the minimum of all the

    // perfect cubes from the array

    int mini = Integer.MAX\_VALUE;

    // Traverse all elements in the array

    for (int i = 0; i < n; i++)

    {

        // Store the minimum if current

        // element is a perfect cube

        if (checkPerfectcube(a[i]))

        {

            mini = Math.min(a[i], mini);

        }

    }

    return mini;

}

// Driver code

public static void main (String[] args)

{

    // int a[] = { 16, 8, 25, 2, 3, 10 };

     Scanner sc = new Scanner(System.in);

         int len= sc.nextInt();

         int a[] = new int[len];

         for(int j=0;j<len;j++)

         {

             a[j]=sc.nextInt();

         }

    int n = a.length;

    System.out.print(smallestPerfectCube(a, n));

}

}

Q.20

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    public static void main(String[] args) {

                // int numbers[] = new int[]{33,53,73,94,22,45,23,87,13,63};

                Scanner sc = new Scanner(System.in);

         int len= sc.nextInt();

         int numbers[] = new int[len];

         for(int j=0;j<len;j++)

         {

             numbers[j]=sc.nextInt();

         }

                int smallest = numbers[0];

                int biggest = numbers[0];

                for(int i=1; i< numbers.length; i++)

                {

                        if(numbers[i] > biggest)

                                biggest = numbers[i];

                        else if (numbers[i] < smallest)

                                smallest = numbers[i];

                }

                System.out.print(biggest);

                System.out.print(" "+smallest);

        }

}

Q.21

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    public static void main(String args[])

  {

    int m, n, c, d;

    Scanner in = new Scanner(System.in);

    // System.out.println("Enter the number of rows and columns of matrix");

    m = 3;

    n = 3;

    int first[][] = new int[m][n];

    int second[][] = new int[m][n];

    int sum[][] = new int[m][n];

    // System.out.println("Enter the elements of first matrix");

    for (c = 0; c < m; c++)

      for (d = 0; d < n; d++)

        first[c][d] = in.nextInt();

    // System.out.println("Enter the elements of second matrix");

    for (c = 0 ; c < m; c++)

      for (d = 0 ; d < n; d++)

        second[c][d] = in.nextInt();

    for (c = 0; c < m; c++)

      for (d = 0; d < n; d++)

        sum[c][d] = first[c][d] + second[c][d];  //replace '+' with '-' to subtract matrices

    // System.out.println("Sum of the matrices:");

    for (c = 0; c < m; c++)

    {

      for (d = 0; d < n; d++)

        System.out.print(sum[c][d]+" ");

      System.out.println();

    }

  }

}

Q.22

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main (String[] args)

    {

       // int [] a = {-2, -3, 4, -1, -2, 1, 5, -3};

        Scanner sc=new Scanner(System.in);

        int len=sc.nextInt();

        int a[]=new int[len];

        for(int i=0;i<len;i++)

        {

            a[i]=sc.nextInt();

        }

                System.out.println(maxSubArraySum(a));

    }

    static int maxSubArraySum(int a[])

    {

        int size = a.length;

        int max\_so\_far = Integer.MIN\_VALUE, max\_ending\_here = 0;

        for (int i = 0; i < size; i++)

        {

            max\_ending\_here = max\_ending\_here + a[i];

            if (max\_so\_far < max\_ending\_here)

                max\_so\_far = max\_ending\_here;

            if (max\_ending\_here < 0)

                max\_ending\_here = 0;

        }

        return max\_so\_far;

    }

}

Q.23

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    // static int arr[] = new int[] { 0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1 };

    // Method for maximum amount of water

    static int findWater(int n,int arr[])

    {

        // left[i] contains height of tallest bar to the

        // left of i'th bar including itself

        int left[] = new int[n];

        // Right [i] contains height of tallest bar to

        // the right of ith bar including itself

        int right[] = new int[n];

        // Initialize result

        int water = 0;

        // Fill left array

        left[0] = arr[0];

        for (int i = 1; i < n; i++)

            left[i] = Math.max(left[i - 1], arr[i]);

        // Fill right array

        right[n - 1] = arr[n - 1];

        for (int i = n - 2; i >= 0; i--)

            right[i] = Math.max(right[i + 1], arr[i]);

        // Calculate the accumulated water element by element

        // consider the amount of water on i'th bar, the

        // amount of water accumulated on this particular

        // bar will be equal to min(left[i], right[i]) - arr[i] .

        for (int i = 0; i < n; i++)

            water += Math.min(left[i], right[i]) - arr[i];

        return water;

    }

    // Driver method to test the above function

    public static void main(String[] args)

    {         Scanner sc=new Scanner(System.in);

    int len=sc.nextInt();

     int arr[]=new int[len];

    for(int i=0;i<len;i++)

    {

        arr[i]=sc.nextInt();

    }

        System.out.println(findWater(len,arr));

    }

}

Q.24

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

   static String decode(String str)

    {

        Stack<Integer> integerstack = new Stack<>();

        Stack<Character> stringstack = new Stack<>();

        String temp = "", result = "";

        // Traversing the string

        for (int i = 0; i < str.length(); i++)

        {

            int count = 0;

            // If number, convert it into number

            // and push it into integerstack.

            if (Character.isDigit(str.charAt(i)))

            {

                while (Character.isDigit(str.charAt(i)))

                {

                    count = count \* 10 + str.charAt(i) - '0';

                    i++;

                }

                i--;

                integerstack.push(count);

            }

            // If closing bracket ']', pop elemment until

            // '[' opening bracket is not found in the

            // character stack.

            else if (str.charAt(i) == ']')

            {

                temp = "";

                count = 0;

                if (!integerstack.isEmpty())

                {

                    count = integerstack.peek();

                    integerstack.pop();

                }

                while (!stringstack.isEmpty() && stringstack.peek()!='[' )

                {

                    temp = stringstack.peek() + temp;

                    stringstack.pop();

                }

                if (!stringstack.empty() && stringstack.peek() == '[')

                    stringstack.pop();

                // Repeating the popped string 'temo' count

                // number of times.

                for (int j = 0; j < count; j++)

                    result = result + temp;

                // Push it in the character stack.

                for (int j = 0; j < result.length(); j++)

                    stringstack.push(result.charAt(j));

                result = "";

            }

            // If '[' opening bracket, push it into character stack.

            else if (str.charAt(i) == '[')

            {

                if (Character.isDigit(str.charAt(i-1)))

                    stringstack.push(str.charAt(i));

                else

                {

                    stringstack.push(str.charAt(i));

                    integerstack.push(1);

                }

            }

            else

                stringstack.push(str.charAt(i));

        }

        // Pop all the elmenet, make a string and return.

        while (!stringstack.isEmpty())

        {

            result = stringstack.peek() + result;

            stringstack.pop();

        }

        return result;

    }

    // Driver method

    public static void main(String args[])

    {

      Scanner sc = new Scanner(System.in);

        String str = sc.nextLine();

        System.out.println(decode(str));

    }

}

Q.25

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    static boolean pairWiseConsecutive(Stack<Integer> s)

{

    // Transfer elements of s to aux.

    Stack<Integer> aux = new Stack<Integer> ();

    while (!s.isEmpty()) {

        aux.push(s.peek());

        s.pop();

    }

    // Traverse aux and see if

    // elements are pairwise

    // consecutive or not. We also

    // need to make sure that original

    // content is retained.

    boolean result = true;

    while (aux.size() > 1) {

        // Fetch current top two

        // elements of aux and check

        // if they are consecutive.

        int x = aux.peek();

        aux.pop();

        int y = aux.peek();

        aux.pop();

        if (Math.abs(x - y) != 1)

        result = false;

        // Push the elements to original

        // stack.

        s.push(x);

        s.push(y);

    }

    if (aux.size() == 1)

        s.push(aux.peek());

    return result;

}

// Driver program

public static void main(String[] args)

{

    Scanner sc = new Scanner(System.in);

  boolean b=true;

    Stack<Integer> s = new Stack<Integer> ();

    int ele=sc.nextInt();

    for(int i=0;i<ele;i++)

    {

        s.push(ele);

    }

    // s.push(1);

    // s.push(2);

    // s.push(6);

    // s.push(7);

    // s.push(34);

    // s.push(35);

    if (pairWiseConsecutive(s))

      b=true;

        System.out.print(b);

    //else

        //System.out.println("No");

    //System.out.println("Stack content (from top) after function call");

    while (s.isEmpty() == false)

    {

    //System.out.print(s.peek() + " ");

    s.pop();

    }

}

}

Q.27

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

        public static int[] removeTheElement(int[] arr,

                                          int index)

    {

        // If the array is empty

        // or the index is not in array range

        // return the original array

        if (arr == null

            || index < 0

            || index >= arr.length) {

            return arr;

        }

        // Create another array of size one less

        int[] anotherArray = new int[arr.length - 1];

        // Copy the elements except the index

        // from original array to the other array

        for (int i = 0, k = 0; i < arr.length; i++) {

            // if the index is

            // the removal element index

            if (i == index) {

                continue;

            }

            // if the index is not

            // the removal element index

            anotherArray[k++] = arr[i];

        }

        // return the resultant array

        return anotherArray;

    }

    // Driver Code

    public static void main(String[] args)

    {

        // Get the array

        // int[] arr = { 1, 2, 3, 4, 5 };

         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();

       }

        // Print the resultant array

        // // System.out.println("Original Array: "

        //                    + Arrays.toString(arr));

        // Get the specific index

        int index = sc.nextInt();

        // Print the index

        // System.out.println(index);

        // Remove the element

        arr = removeTheElement(arr, index);

        // Print the resultant array

        for(int j=0;j<arr.length;j++)

        {

            System.out.print(arr[j]);

            System.out.print(" ");

        }

       // System.out.println(Arrays.toString(arr));

    }

}

Q.28

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String[] args) {

        //Initialize array

        // int [] arr = new int [] {1, 2, 3, 4, 5};

        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();

        }

        // System.out.println();

        //Loop through the array in reverse order

        for (int i = arr.length-1; i >= 0; i--) {

            System.out.print(arr[i]+ " ");

        }

    }

}

Q.29

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String args[]) {

    Scanner sc=new Scanner(System.in);

    String str=sc.nextLine();

        System.out.println(isPalindromString(str));

    }

    /\*\*

     \* Java method to check if given String is Palindrome

     \* @param text

     \* @return true if text is palindrome, otherwise false

     \*/

    public static boolean isPalindromString(String text){

       String reverse = reverse(text);

       if(text.equals(reverse)){

           return true;

       }

       return false;

    }

    /\*\*

     \* Java method to reverse String using recursion

     \* @param input

     \* @return reversed String of input

     \*/

    public static String reverse(String input){

        if(input == null || input.isEmpty()){

            return input;

        }

        return input.charAt(input.length()- 1) + reverse(input.substring(0, input.length() - 1));

    }

}

Q.30

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

   static boolean arePermutation(String str1, String str2)

{

    // Get lenghts of both strings

    int n1 = str1.length();

    int n2 = str2.length();

    // If length of both strings is not same,

    // then they cannot be Permutation

    if (n1 != n2)

    return false;

    char ch1[] = str1.toCharArray();

    char ch2[] = str2.toCharArray();

    // Sort both strings

    Arrays.sort(ch1);

    Arrays.sort(ch2);

    // Compare sorted strings

    for (int i = 0; i < n1; i++)

    if (ch1[i] != ch2[i])

        return false;

    return true;

}

/\* Driver program to test to print printDups\*/

public static void main(String[] args)

{

    Scanner sc = new Scanner(System.in);

    boolean b=true;

    String str1 = sc.nextLine();

    String str2 = sc.nextLine();

    if (arePermutation(str1, str2))

    {

      b=true;

    System.out.print(b);

    }

    else

    {

      b=false;

    System.out.print(b);

    }

}

}

Q.32

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

   static int countOccurrences(int arr[], int n, int x)

    {

        int res = 0;

        for (int i=0; i<n; i++)

            if (x == arr[i])

              res++;

        return res;

    }

    public static void main(String args[])

    {

        Scanner sc =new Scanner(System.in);

        int len = sc.nextInt();

        int arr[]=new int[len];

        for(int i=0;i<len;i++)

        {

            arr[i]=sc.nextInt();

        }

        //int arr[] = {1, 2, 3, 4, 5};

        int n = arr.length;

        int x = sc.nextInt();

        System.out.println(countOccurrences(arr, n, x));

    }

}

Q.33

import java.util.Scanner;

// Java implementaion of the approach

class Solution {

    // Represents a node of linked list

    static class Node {

        int data;

        Node next;

    }

    // Function to insert node in a linked list

    static Node insert(Node head, int item)

    {

        Node ptr = head;

        Node temp = new Node();

        temp.data = item;

        temp.next = null;

        if (head == null)

            head = temp;

        else {

            while (ptr.next != null)

                ptr = ptr.next;

            ptr.next = temp;

        }

        return head;

    }

    // Function to find the sum of non duplicate nodes

    static int sumOfNonDupNode(Node head)

    {

        Node ptr1 = head;

        Node ptr2;

        int sum = 0;

        while (ptr1 != null) {

            ptr2 = head;

            boolean flag = false;

            // Check if current node has some duplicate

            while (ptr2 != null) {

                // Check for duplicate node

                if (ptr1 != ptr2 && ptr1.data == ptr2.data) {

                    flag = true;

                    break;

                }

                // Get to the next node

                ptr2 = ptr2.next;

            }

            // If current node is unique

            if (!flag)

                sum += ptr1.data;

            // Get to the next node

            ptr1 = ptr1.next;

        }

        return sum;

    }

    // Driver code

    public static void main(String args[])

    { Scanner sc = new Scanner(System.in);

        //Solution s1 = new Solution();

        Node head = null;

        // head = insert(head, 1);

        // head = insert(head, 1);

        // head = insert(head, 3);

        // head = insert(head, 4);

        // head = insert(head, 5);

        int len=sc.nextInt();

        for(int i=0;i<len;i++)

        {

            int num=sc.nextInt();

            head=insert(head, num);

        }

        System.out.print(Solution.sumOfNonDupNode(head));

    }

}

Q.37

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

import java.util.Queue;

public class Solution {

   public static void main(String[] args) {

       Scanner sc= new Scanner(System.in);

        int len=sc.nextInt();

    Queue<Integer> queue = new LinkedList<>();

    for(int i=1;i<=len;i++)

    {

        int num=sc.nextInt();

        queue.add(num);

    }

    // queue.add("two");

    // queue.add("three");

    // queue.add("four");

    // // System.out.println(queue);

     boolean b=true;

    queue.remove(0);

    // System.out.println(queue);

    // System.out.println("Queue Size: " + queue.size());

    //System.out.println(queue.contains(0));

    if(queue.contains(0))

    {

        b=true;

        System.out.println(b);

    }

    else

    {

        b=false;

        System.out.println(b);

    }

    // To empty the queue

    // queue.clear();

   }

}

Q.38

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

   static boolean pairWiseConsecutive(Queue<Integer> q)

{

    // Transfer elements of q to aux.

    Stack<Integer> aux = new Stack<>();

    while (!q.isEmpty()) {

        aux.push(q.peek());

        q.poll();

    }

    // Again transfer the

    // elements of aux to aux2

    Stack<Integer> aux2 = new Stack<>();

    while (!aux.empty()) {

        aux2.push(aux.peek());

        aux.pop();

    }

    // Traverse aux2 and see if

    // elements are pairwise

    // consecutive or not. We also

    // need to make sure that original

    // content is retained.

    boolean result = true;

    while (aux2.size() > 1) {

        // Fetch current top two

        // elements of aux2 and check

        // if they are consecutive.

        int x = aux2.peek();

        aux2.pop();

        int y = aux2.peek();

        aux2.pop();

        if (Math.abs(x - y) != 1)

            result = false;

        // Push the elements to queue

        q.add(x);

        q.add(y);

    }

    if (aux2.size() == 1)

        q.add(aux2.peek());

    return result;

}

// Driver program

static public void main(String[] args) {

        // Pushing elements into the queue

        Scanner sc= new Scanner(System.in);

        int len= sc.nextInt();

    Queue<Integer> q= new LinkedList<Integer>();

    for (int i=0;i<len;i++)

    {

        int num=sc.nextInt();

        q.add(num);

    }

    // q.add(11);

    // q.add(12);

    // q.add(13);

    // q.add(14);

    boolean b=true;

    if (pairWiseConsecutive(q))

    {

       b=true;

        System.out.println(b);

    }

    else

    {

      b=false;

        System.out.println(b);

    }

    // Printing the original queue

    while (!q.isEmpty()) {

        //System.out.print(q.peek() + " ");

        q.remove();

    }

    //System.out.println();

    }

}

Q.39

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

   public static void main(String args[])

{

    Scanner sc = new Scanner(System.in);

 int n1=0,n2=1,n3,i;

 int count=sc.nextInt();

 System.out.print(n1+" "+n2);//printing 0 and 1

 for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

 {

  n3=n1+n2;

  System.out.print(" "+n3);

  n1=n2;

  n2=n3;

 }

}}

Q.41

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

 static final int MOD = 1000000007;

    public static long countBT(int h) {

        long[] dp = new long[h + 1];

        // base cases

        dp[0] = 1;

        dp[1] = 1;

        for(int i = 2; i <= h; ++i)

            dp[i] = (dp[i - 1] \* ((2 \* dp [i - 2])% MOD + dp[i - 1]) % MOD) % MOD;

            return dp[h];

    }

    // Driver program

    public static void main (String[] args) {

        Scanner sc =new Scanner(System.in);

        int h = sc.nextInt();

        System.out.println(countBT(h));

    }