Given an array of N elements where each element is either 1 or 0. You have to divide the array into maximum number of subarrays such that each element of the array is in exactly one subarray such that each subarray has equal number of 1's and 0's.

Input

First line of input contains N.  
Second line of input contains N space separated elements of the array.  
  
Constraints:  
1 <= N <= 100000  
0 <= elements of the array <= 1

Output

Print the single integer which is the maximum number of subarrays the array can be divided into. If it is not possible then print -1.

import java.io.\*; // for handling input/output

import java.util.\*; // contains Collections framework

// don't change the name of this class

// you can add inner classes if needed

class Main {

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int[] arr=new int[n];

        int c0=0,c1=0;

        int count=0;

        for(int i=0;i<n;i++)

        {

            arr[i]=sc.nextInt();

            if(arr[i]==1)

            c1++;

            if(arr[i]==0)

            c0++;

            if(c1==c0)

            count++;

        }

        if(c1==c0){

            System.out.print(count);

        }

        else{

            System.out.print(-1);

        }

        }

}

Given a positive integer N, construct a string of length N such that no two adjacent characters are same. Among all possible strings, find the lexicographically minimum string.  
Note: You can use only lowercase characters from 'a' to 'z'.

import java.io.\*; // for handling input/output

import java.util.\*; // contains Collections framework

// don't change the name of this class

// you can add inner classes if needed

class Main {

    public static void main (String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

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

                  if(i%2 == 0)

                    System.out.print("a");

                  else

                    System.out.print("b");

                }

    }

}

Given an array of N integers containing only 0 or 1. You can do following operations on the array:  
> swap elements at two indices.  
> choose one index and change its value from 0 to 1 or vice- versa.  
You have to do minimum number of above operations such that the final array is non-decreasing.

Input

First line of input contains a single integer N.  
Second line of input contains N space separated integers denoting the array.  
  
Constraints:  
1 <= N <= 100000  
elements of the array are 0 or 1.

Output

Minimum number of moves required such that the final array is non- decreasing.

import java.io.\*; // for handling input/output

import java.util.\*; // contains Collections framework

// don't change the name of this class

// you can add inner classes if needed

class Main {

    static void swap(int a[],int i,int j)

    {

        int temp =a[i];

        a[i]=a[j];

        a[j]=temp;

    }

    public static void main (String[] args) {

        Scanner sc = new Scanner(System.in);

        int N = sc.nextInt();

        int []a = new int[N];

        for(int i=0;i<N;i++){

             a[i]=sc.nextInt();

            }

        int i=0;

        int j=a.length-1;

        int count=0;

        while(i<j){

            while(a[i]==0) i++;

            while(a[j]==1) j--;

                if(i<j){

                    swap(a,i,j);

                    count++;

                }

            }

            System.out.println(count);

    }

}

Sara's brother gives her a string representing chocolates and candies in a row. Sara loves eating candies before chocolates so she wants to rearrange the given string so that all the candies come before chocolates. Help her to rearrange the string.  
You are allowed only 1 type of operation as many times as you want. You can swap all existing pair of candy and chocolate in a second if they are consecutive and chocolate is before candy, that means in one second you can swap all the i-th value with i+1 if and only if the i-th position has chocolate and the i+1 has candy.  
Given the String S, find the minimum time needed to rearrange the given string.

Input

Input contains a single line containing the string S.  
  
Constraints:-  
1 <= |S| <= 1000000  
  
Note:- String will contain only characters 'M' and 'F' where M represents chocolate while F represents candy.

Output

Print the minimum time needed to rearrange the given string such that all candies come before chocolates.

import java.io.\*; // for handling input/output

import java.util.\*; // contains Collections framework

// don't change the name of this class

// you can add inner classes if needed

class Main {

    public static void main (String[] args) {

       Scanner sc=new Scanner(System.in);

       String s=sc.next();

       int t=0,c=0;

       for(int i=0;i<s.length();i++){

            if(s.charAt(i)=='F'){

                if((i-c)<=t && t!=0){

                    t++;

                }

                else{

                    t=i-c;

                }

                c++;

            }

        }

        System.out.println(t);

    }

}

You have bought N cakes and the ith cake has A[i] level of sweetness. Now you want to display the cakes in exhibition but there is a condition that you cannot display two cakes with same sweetness level. Find the maximum number of cakes that you can display.

import java.io.\*; // for handling input/output

import java.util.\*; // contains Collections framework

// don't change the name of this class

// you can add inner classes if needed

class Main {

    static int countDistinct(long arr[], int n){

        int res = 0;

        int i;

        int j;

        for(i=0;i<n;i++){

            for(j=0;j<n;j++)

            if(arr[i] == arr[j])

            break;

            if(i==j)

            res++;

        }

        return res;

        }

        public static void main(String[] args){

            Scanner sc=new Scanner(System.in);

            int n=sc.nextInt();

            long arr[]=new long[n];

            for(int i=0; i<n; i++){

                arr[i]=sc.nextLong();

            }

            System.out.println(countDistinct(arr,n));

        }

}

JAVA QUESTIONS AND THEORY

1. Class main{

Public static void main(string args[]) {

System.out.println(“hello world!”);

}

}