

/\*Problem Statement – An automobile company manufactures both a two wheeler (TW) and a four wheeler (FW). A company manager wants to make the production of both types of vehicle according to the given data below:

1st data, Total number of vehicle (two-wheeler + four-wheeler)=v

2nd data, Total number of wheels = W

The task is to find how many two-wheelers as well as four-wheelers need to manufacture as per the given data.

Example :

Input :

200 -> Value of V

540 -> Value of W

Output :

TW =130 FW=70

Explanation:

$130+70 = 200$  vehicles

$(70*4)+(130*2)= 540$  wheels

Constraints :

$2 \leq W$

$W \% 2 = 0$

$V < W$

Print "INVALID INPUT" , if inputs did not meet the constraints.

The input format for testing

Vaibhav Lanjewar

The candidate has to write the code to accept two positive numbers separated by a new line.

First Input line – Accept value of V.

Second Input line- Accept value for W.

The output format for testing

Written program code should generate two outputs, each separated by a single space character(see the example)

Additional messages in the output will result in the failure of test case \*/

```
import java.util.*;

public class q1{

    public static void main(String args[]){

        Scanner sc = new Scanner(System.in);

        int v=sc.nextInt();

        int w=sc.nextInt();

        float t=((4*v)-w)/2;

        float f=(w-2*v)/2;

        if(w>=2 && w%2==0 && w>v){

            System.out.println( "TF= "+(int)t+" FW= "+(int)f);

        }

        else{

            System.out.println("INVALID INPUT");

        }

    }

}
```

```
//-----
```

/\*Problem Statement – Given a string S(input consisting) of '\*' and '#'. The length of the string is variable. The task is to find the minimum number of '\*' or '#' to make it a valid string. The string is considered valid if the number of '\*' and '#' are equal. The '\*' and '#' can be at any position in the string.

Note : The output will be a positive or negative integer based on number of '\*' and '#' in the input string.

(\*>#): positive integer

(#>\*): negative integer

(#=\*): 0

Example 1:

Input 1:

###\*\*\* -> Value of S

Output :

0 → number of \* and # are equal \*/

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

```
public class q2 {
```

```
public static void main(String args[]){
```

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

```
    // input string
```

```
    String str=sc.nextLine();
```

```
    int cnt_star=0;
```

```
    int cnt_hash=0;
```

```

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

    if(str.charAt(i)=='*'){

        cnt_star++;

    }

    else if(str.charAt(i)=='#'){

        cnt_hash++;

    }

}

int ans=cnt_star-cnt_hash;

System.out.println(ans);

}

}

```

//-----VNBL-----

/\*Given an integer array Arr of size N the task is to find the count of elements whose value is greater than all of its prior elements.

Note : 1st element of the array should be considered in the count of the result.

For example,

Arr[]={7,4,8,2,9}

As 7 is the first element, it will consider in the result.

8 and 9 are also the elements that are greater than all of its previous elements.

Since total of 3 elements is present in the array that meets the condition.

Hence the output = 3.

Example 1:

Input

5 -> Value of N, represents size of Arr

7-> Value of Arr[0]

4 -> Value of Arr[1]

8-> Value of Arr[2]

2-> Value of Arr[3]

9-> Value of Arr[4]

Output :

3

Example 2:

5 -> Value of N, represents size of Arr

3 -> Value of Arr[0]

4 -> Value of Arr[1]

5 -> Value of Arr[2]

8 -> Value of Arr[3]

9 -> Value of Arr[4]

Output :

5

Constraints

$1 \leq N \leq 20$

$1 \leq \text{Arr}[i] \leq 10000$

\*/

```
import java.util.*;

public class q3{

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        int size=sc.nextInt();

        int arr[]=new int[size];

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

            arr[i]=sc.nextInt();

        }

        int cnt=1;

        for(int i=1;i<size;i++){

            if(arr[i-1]<arr[i]){

                cnt++;

            }

        }

        System.out.println(cnt);

    }

}

//-----
```

/\* A parking lot in a mall has RxC number of parking spaces. Each parking space will either be empty(0) or full(1). The status (0/1) of a parking space is represented as the element of the matrix. The task is to find index of the prpeinzta row(R) in the parking lot that has the most of the parking spaces full(1).

Note :

RxC- Size of the matrix

Elements of the matrix M should be only 0 or 1.

Example 1:

Input :

3 -> Value of R(row)

3 -> value of C(column)

[0 1 0 1 1 0 1 1 1] -> Elements of the array M[R][C] where each element is separated by new line.

Output :

3 -> Row 3 has maximum number of 1's

Example 2:

input :

4 -> Value of R(row)

3 -> Value of C(column)

[0 1 0 1 1 0 1 0 1 1 1] -> Elements of the array M[R][C]

Output :

4 -> Row 4 has maximum number of 1's\*/

// <https://prepinsta.com/tcs-coding-questions/>

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

```
public class q4 {
```

Vaibhav Lanjewar

```

public static void main(String args[]){

    Scanner sc=new Scanner(System.in);

    int row=sc.nextInt();

    int col=sc.nextInt();


    int[][]arr=new int[row][col];

    int max=0;

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

        for(int j=0;j<col;j++){

            arr[i][j]=sc.nextInt();


        }

    }


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

        for(int j=0;j<col;j++){

            if(arr[i][j]==1){

                max=Math.max(max,i);

            }

        }

    }

    System.out.println("Row " + row + "Has max number of 1's");

}

}

//-----

```

/\*TCS Coding Question Day 2 Slot 1 – Question 1



A party has been organised on cruise. The party is organised for a limited time(T). The number of guests entering (E[i]) and leaving (L[i]) the party at every hour is represented as elements of the array. The task is to find the maximum number of guests present on the cruise at any given instance within T hours.

// <https://prepinsta.com/tcs-coding-questions/>

Example 1:

Input :

5 -> Value of T

[7,0,5,1,3] -> E[], Element of E[0] to E[N-1], where input each element is separated by new line

[1,2,1,3,4] -> L[], Element of L[0] to L[N-1], while input each element is separate by new line.

Output :

8 -> Maximum number of guests on cruise at an instance.

Explanation:

1st hour:

Entry : 7 Exit: 1

No. of guests on ship : 6

2nd hour :

Entry : 0 Exit : 2

No. of guests on ship :  $6-2=4$

Hour 3:

Entry: 5 Exit: 1

No. of guests on ship :  $4+5-1=8$

Hour 4:

Vaibhav Lanjewar

Entry : 1 Exit : 3

No. of guests on ship :  $8+1-3=6$

Hour 5:

Entry : 3 Exit: 4

No. of guests on ship:  $6+3-4=5$

Hence, the maximum number of guests within 5 hours is 8.

Example 2:

Input:

4 -> Value of T

[3,5,2,0] -> E[], Element of E[0] to E[N-1], where input each element is separated by new line.

[0,2,4,4] -> L[], Element of L[0] to L[N-1], while input each element in separated by new line

Output:

6

Cruise at an instance

Explanation:

Hour 1:

Entry: 3 Exit: 0

No. of guests on ship: 3

Hour 2:

Entry : 5 Exit : 2

No. of guest on ship:  $3+5-2=6$

Hour 3:

Entry : 2 Exit: 4

No. of guests on ship:  $6+2-4=4$

Hour 4:

Entry: 0 Exit : 4

No. of guests on ship :  $4+0-4=0$

Hence, the maximum number of guests within 5 hours is 6.

The input format for testing

The candidate has to write the code to accept 3 input.

First input- Accept value for number of T(Positive integer number)

Second input- Accept T number of values, where each value is separated by a new line.

Third input- Accept T number of values, where each value is separated by a new line.

The output format for testing

The output should be a positive integer number or a message as given in the problem statement(Check the output in Example 1 and Example 2)

Constraints:

$1 \leq T \leq 25$

$0 \leq E[i] \leq 500$

$0 \leq L[i] \leq 500$

\*/

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

```
public class q5 {
```

```
    public static void main(String args[]){
```

```

Scanner sc=new Scanner(System.in);

int size=sc.nextInt();

int []E=new int[size];

int []L=new int[size];


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

E[i]=sc.nextInt();


}

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

    L[i]=sc.nextInt();

}


int maxGeuest=0;

int sum=0;

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

sum+=E[i]-L[i];

maxGeuest=Math.max(maxGeuest,sum);

}

System.out.println("Maximun number of geuest are "+maxGeuest);


}

}

//-----

```

### /\*TCS Coding Question Day 2 Slot 1 – Question 2

At a fun fair, a street vendor is selling different colours of balloons. He sells N number of different colours of balloons (B[]). The task is to find the colour (odd) of the balloon which is present odd number of times in the bunch of balloons.

Note: If there is more than one colour which is odd in number, then the first colour in the array which is present odd number of times is displayed. The colours of the balloons can all be either upper case or lower case in the array. If all the inputs are even in number, display the message “All are even”.

Example 1:

7 -> Value of N

[r,g,b,b,g,y,y] -> B[] Elements B[0] to B[N-1], where each input element is separated by new line.

Output :

r -> [r,g,b,b,g,y,y] -> “r” colour balloon is present odd number of times in the bunch.

Explanation:

From the input array above:

r: 1 balloon

g: 2 balloons

b: 2 balloons

y : 2 balloons

Hence , r is only the balloon which is odd in number.

Example 2:

Input:

10 -> Value of N

[a,b,b,b,c,c,c,a,f,c] -> B[], elements B[0] to B[N-1] where input each element is separated by new line.

Output :

b-> 'b' colour balloon is present odd number of times in the bunch.

Explanation:

From the input array above:

a: 2 balloons

b: 3 balloons

c: 4 balloons

f: 1 balloons

Here, both 'b' and 'f' have odd number of balloons. But 'b' colour balloon occurs first.

Hence , b is the output.

Input Format for testing

The candidate has to write the code to accept: 2 input

First input: Accept value for number of N(Positive integer number).

Second Input : Accept N number of character values (B[]), where each value is separated by a new line.

Output format for testing

The output should be a single literal (Check the output in example 1 and example 2)

Constraints:

$3 \leq N \leq 50$

$B[i] = \{ \{a-z\} \text{ or } \{A-Z\} \}^*$

```
import java.util.*;

public class q6{

    public static void main(String args[]){

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

        // int n=sc.nextInt();

        // char[]color=new char[n];

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

        //     color[i]=sc.next().charAt(0); //input char to char array sc.next().charAt(0);

        // }

        // HashMap<Character,Integer>mp=new HashMap<>();

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

        //     char v=color[i];

        //     mp.put(color[i],mp.getOrDefault(color[i],0)+1);

        // }

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

        //     if(mp.get(color[i])%2==1){

        //         System.out.println("odd ballon is "+color[i]+ " has count is "+mp.get(color[i]));

        //         break;

        //     }

        // }
```

```
}
```

```
}
```

```
//-----
```

/\*A chocolate factory is packing chocolates into the packets. The chocolate packets here represent an array of N number of integer values. The task is to find the empty packets(0) of chocolate and push it to the end of the conveyor belt(array).

Example 1 :

N=8 and arr = [4,5,0,1,9,0,5,0].

There are 3 empty packets in the given set. These 3 empty packets represented as 0 should be pushed towards the end of the array

Input :

8 – Value of N

[4,5,0,1,9,0,5,0] – Element of arr[0] to arr[N-1], While input each element is separated by newline

Output:

4 5 1 9 5 0 0 0

\*/

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

```
public class q7 {
```

Vaibhav Lanjewar



```
public static void main(String args[]){

    Scanner sc=new Scanner(System.in);

    int n=sc.nextInt();

    int choco[]=new int[n];

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

        choco[i]=sc.nextInt();

    }

    int j=0;

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

        if(choco[i]!=0){

            choco[j++]=choco[i];

        }

    }

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

        choco[i]=0;

    }

    for(int i:choco){

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

    }

}

//-----
```

/\*TCS NQT Coding Question 2023 – September Day 1 – Slot 1

Problem Statement –

Joseph is learning digital logic subject which will be for his next semester. He usually tries to solve unit assignment problems before the lecture. Today he got one tricky question. The problem statement is “A positive integer has been given as an input. Convert decimal value to binary representation. Toggle all bits of it after the most significant bit including the most significant bit. Print the positive integer value after toggling all bits”.

Constrains-

$1 \leq N \leq 100$

Example 1:

Input :

10 -> Integer

Output :

5 -> result- Integer

Explanation:

Binary representation of 10 is 1010. After toggling the bits(1010), will get 0101 which represents “5”. Hence output will print “5”. \*/

```
import java.util.*;

public class q8 {

    public static void main(String args[]){

        Scanner sc = new Scanner(System.in);

        int n=sc.nextInt();

        StringBuilder sb =new StringBuilder();

        while(n>0){

            int a=0;

            a=n&1;

            if(a==0){

                sb.insert((char)1,0);

            }else{

                sb.insert((char)0,0);

            }

            n=n>>1;

        }

        System.out.println(Integer.parseInt(sb.toString()));

    }

}
```

```
//-----
```

```
import java.util.Scanner;
```

```
public class q9 {
```

```
    // gcd/hcf
```

```
    public static void main(String args[]){
```

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

```
        int a=sc.nextInt();
```

```
        int b=sc.nextInt();
```

```
        if(a==0){
```

```
            System.out.println(b);
```

```
            return ;
```

```
        }
```

```
        else if(b==0){
```

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

```
            return ;
```

```
        }
```

```
        while(a!=b){
```

```
            if(a>b){
```

```
                a=a-b;
```

```
            }
```

```
            else{
```

```
                b=b-a;
```

```
            }
```

```

        if(a==b){

            System.out.println(a);

        }

    }

}

}

//-----

```

/\*Q1. A carpet manufacturing industry has newly ventured into the carpet installation business. All the carpets

manufactured are large squares in shape. To install, each carpet has to be cut into shapes of squares or

rectangles. The number of slits to be made is given as N.

The task is to find the maximum number of equal squares or rectangles that can be achieved using N slits.

Note:

The square carpet can be cut only using horizontal or vertical slits.

Cuttings are done on a single carpet which should be rolled out completely i.e. no folding or stacking is

allowed.

Squares or rectangles cut should be equal size.

Example 1:

Input:

4 → Value of N(No. of cuts)

Output:

9 → maximum number of equal squares or rectangles

Explanation:

Solution 2

Maximum number of squares/rectangles that can be obtained with N=4 is 9(Solution 1)

Hence, output is 9

Example 2:

Input:

1 → Value of N(No. of teams)

Output:

2 → maximum number of equal squares or rectangles \*/

```
import java.util.*;

public class q10 {

    //vnbl

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int x=n/2;

        System.out.println( ((x+1)*(n-x+1)));

    }

}

//-----

/*

Code
```

## 650. 2 Keys Keyboard

// <https://leetcode.com/problems/2-keys-keyboard/description/?envType=daily-question&envId=2024-08-19>

There is only one character 'A' on the screen of a notepad. You can perform one of two operations on this notepad for each step:

Copy All: You can copy all the characters present on the screen (a partial copy is not allowed).

Paste: You can paste the characters which are copied last time.

Given an integer  $n$ , return the minimum number of operations to get the character 'A' exactly  $n$  times on the screen. \*/

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

```
public class q11 {
```

```
    public static int step(int n){
```

```
        int s=0;
```

```
        for(int i=2;i<=n;i++){
```

```
            while(n%i==0){
```

```
                s+=i;
```

```
                n=n/i;
```

```
            }
```

```
        }
```

```
        return s;
```

```
    }
```

```
    public static void main(String[] args) {
```

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

```
        int n=sc.nextInt();
```

```
        System.out.println(step(n));
```

```
    }  
}  
//-----
```

/\*Q2. Given an array Arr[] of size T, contains binary digits.

Where

0 represents a biker running to the north.

1 represents a biker running to the south.

The task is to count crossing bikers in such a way that each pair of crossing bikers (N, S), where  $0 \leq N < S < T$ ,

is passing when N is running to the north and S is running to the south.

Constraints:

$0 \leq N < S < T$

Example -1:

Input:

5.-> Number of elements i.e. T

0.-> Value of 1st element

1.-> Value of 2nd element

0.-> Value of 3rd element

1.-> Value of 4th element

1.-> Value of 5th element

Output:

5

Explanation:

The 5 pairs are (Arr[0], Arr[1]), (Arr[0], Arr[3]), (Arr[0], Arr[4]), (Arr[2], Arr[3]) and (Arr[2], Arr[4]).



Note that in all pairs first element is 0, second element is 1 and index of first element is smaller than index

of second element.

The Input format for testing:

First input line: Accept a single positive integer value for T representing the size of Arr[].

Second input line:: Accept N number of integer values (0 or 1) separated by a new line.

Output Format for Testing:

The output must be a non-negative integer number only (See the output format in example).

Additional messages in the output will result in the failure of test cases.

Code Solution

```
#include <iostream>

using namespace std;

int main()
{
    int n, count=0;

    25

    cin>>n;

    int B[n];

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

    cin>>B[i];

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

    {

    if(B[i]==0)

    {

    for(int k=i+1; k<n; k++)

    {
```

```

if(B[k]==1)

count++;

}

}

}

cout<<count;

}

*/

```

// my solution

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

```
public class q12_01_ns_pair {
```

```
    public static int pair(int[] dir) {
```

```
        int n = dir.length;
```

```
        int cnt = 0;
```

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

```
            for (int j = i + 1; j < n; j++) {
```

```
                if (dir[i] == 0 && dir[j] == 1 && i < j) {
```

```
                    cnt++;
```

```
                }
```

```
            }
```

```

    }

    return cnt;

}

public static void main(String args[]) {

    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int dir[] = new int[n];

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

        dir[i] = sc.nextInt();

    }

    System.out.println(pair(dir));

}

}

//-----

```

/\*Q1. Airport security officials have confiscated several items of the passenger at the security checkpoint. All

the items have been dumped into a huge box(array). Each item possessed a certain amount of risk(0,1,2).

Here is the risk severity of the item representing an array[] of N number of integer values. The risk here is to

sort the item based on their level of risk values range from 0 to 2.

Example 1:

Vaibhav Lanjewar

Input:

7 ----- Value of N

[1,0,2,0,1,0,2] -> Element of arr[0] to arr[N-1], while input each element is separated by new line

Output:

0 0 0 1 1 2 2 -> Element after sorting based on the risk severity.

Example 2:

Input:

10 ----- Value of N

[2,1,0,2,1,0,0,1,2,0] -> Element of arr[0] to arr[N-1], while input each element is separated by new line

Output:

0 0 0 0 0 1 1 1 2 2 2 -> Element after sorting based on the risk severity.

Constraints

$0 < N \leq 100$

$0 \leq \text{arr}[i] \leq 2$  \*/

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

```
public class q13_airport_sort_color {
```

```
    public static void swap(int nums[],int i,int j){
```

```
        int temp=nums[i];
```

```
        nums[i]=nums[j];
```

```
        nums[j]=temp;
```

```
    }
```

```
    public static void sort(int nums[]){
```

```
        int index=0,end=nums.length-1,start=0;
```

```
while(index<=end){  
    if(nums[index]==0){  
        swap(nums,index,start);  
        index++;  
        start++;  
    }  
}
```

```
else if( nums[index]==2 ){  
    swap(nums,index,end);  
    end--;  
}  
else{  
    index++; //skip 1  
}  
  
}
```

```
}
```

```
public static void main(String args[]){
```

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

```
    int n=sc.nextInt();
```

```
    int nums[]=new int [n];
```

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

```
        nums[i]=sc.nextInt();
```

```

    }

    sort(nums);

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

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

    }

}

}

```

```

/*7
1 0 2 0 1 0 2
0 0 0 1 1 2 2 */

```

```
//-----
```

/\* Q2. An event management company has come up with a unique idea of printing their event tickets. Based

on the ticket number combination (str1), the visitor is directed towards a particular class of audience. The

task is to create a program/application to fetch the ticket number based on the following conditions:

Any occurrences of digits EF, 56 and G, & should be deleted

The characters EF should be in the same format.

Example 1:

Input:

4523EF58G -> Value of STR1

Output:

452358 -> After removal of characters

‘EF’ and ‘G’

Example 2:

Input:

E12F35G58 -> Value of STR1

Output:

E12F3558 -> After removal of character 'G'

Explanation:

In the above example, characters E and F are not together. So, they won't be deleted. The output will be with

only character G removal.

The Input format for testing

The candidate has to write the code to accept 1 input(s).

First input - Accept value for str1 which is a string consisting of numbers and uppercase alphabets without

any

The output format for testing

The output should be a string without any spaces (Check the output in Example 1 and Example 2)

Additional messages in output will cause the failure of test cases.

Constraints:

Str={A,Z),(0-9)}

No spaces and special characters allowed.

Only uppercase alphabets in the input string\*/

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

```
public class q14 {
```

```
    public static void main(String args[]){
```

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

```

String str=sc.next();

StringBuilder sb=new StringBuilder();

char []ch=str.toCharArray();

int n=ch.length;

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

    int j=i+1;

    if(ch[i]=='x') continue;

    if( ch[i]=='E' && ch[i+1]=='F' && j<n){

        ch[i]='x';

        ch[j]='x';

    }

    if(ch[i]=='5' && ch[i+1]=='6' && j<n){

        ch[i]='x';

        ch[j]='x';

    }

    if(ch[i]=='G'){

        ch[i]='x';

    }

    if(ch[i]=='5' && ch[i+1]=='6' && j<n){

        ch[i]='x';

        ch[j]='x';

    }

}

for(char i:ch){

    if(i=='x')continue;

```



```

        else{

            sb.append(i);

        }

    }

    System.out.println(sb.toString());

}

}

//-----

/*

```

Q1. A supermarket maintains a pricing format for all its products. A value N printed on each product. When

the scanner reads the value N on the item, the product of all the digits in the value N is the price of the item.

The task is to design a software such that given the code of any item N the product(multiplication) of all the

digits of value should be computed(price).

Example 1:

Input:

5244 -->Value of N

Output:

160 -->Price

Explanation:

From the input above:

Product of the digits: 5,2,4,4

$5*2*4*4 = 160$

Hence Output is 160

```
*/

import java.util.*;

public class q15_suppermarket_numbers_digit_sum {

    public static int bill(int n){

        //base condition

        if(n/10 ==n) return 1;

        // task

        return (n%10) * bill(n/10);

    }

    // iterative approach

    public static int itrBill(int n){

        int tot=1;

        while(n!=0){

            tot*=n%10;

            n=n/10;

        }

        return tot;

    }

}
```

```
}
```

```
public static void main(String args[]){
```

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

```
    int n=sc.nextInt();
```

```
    //System.out.println(bill(n));
```

```
    System.out.println(itrBill(n));
```

```
}
```

```
}
```

```
//-----
```

/\*Problem Statement:

You are given an integer array nums. The unique elements of an array are the elements that appear exactly once in the array. Your task is to return the sum of all the unique elements in nums.

Example 1:

Input: nums = [1, 2, 3, 2]

Output: 4

Explanation: The unique elements are [1, 3], and their sum is 4.

Example 2:

Vaibhav Lanjewar

Input: nums = [1, 1, 1, 1, 1]

Output: 0

Explanation: There are no unique elements, so the sum is 0.

You need to write a function that takes in the array nums and returns the sum of the unique elements.

```
*/  
  
import java.util.*;  
  
public class q16_sum_of_unique_elements_14_5_24 {  
  
    public static void main(String args[]){  
  
        Scanner sc=new Scanner(System.in);  
  
        int size=sc.nextInt();  
  
        int []nums=new int[size];  
  
  
        HashMap<Integer,Integer>mp=new HashMap<>();  
  
  
        for(int i=0;i<size;i++){  
  
            nums[i]=sc.nextInt();  
  
            mp.put(nums[i],mp.getOrDefault(nums[i],0)+1);  
  
        }  
  
        int sum=0;  
  
        for(int i:nums){  
  
            if(mp.get(i)==1){  
  
                sum+=i;  
  
            }  
  
        }  
  
        System.out.println(sum);  
    }  
}
```

```
}
```

```
}
```

```
//-----
```

```
/*Problem Statement:
```

Problem Statement: kadaness

Given an integer array nums, find the subarray with the largest sum and return its sum.

Example:

Input: nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output: 6

Explanation: The subarray [4, -1, 2, 1] has the largest sum, which is 6.

```
*/
```

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

```
public class q17_Maxsum_of_SubArray_14_5_24 {
```

```
    public static void main(String args[]){
```

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

```
        int size=sc.nextInt();
```

```
        int []nums=new int[size];
```

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

```
            nums[i]=sc.nextInt();
```

```

    }

    int MaxSum=0;

    int currSum=0;

    for(int i:nums){

        currSum+=i;

        if(currSum<0){

            currSum=0;

        }

        MaxSum=Math.max(MaxSum,currSum);

    }

    System.out.println(MaxSum);

}

}

//-----

/*

```

The task is to write a program to calculate the total shipping cost based on the weight of the package and the distance it needs to travel. The shipping cost is determined by the following criteria:

Base money: \$5.00

Cost per kilogram: \$2.00

Cost per 10 kilometers: \$0.50

example :

Vaibhav Lanjewar

Weight (w): 10 kg

Distance (D): 100 km

Output: \$30.00

\*/

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

```
public class q18_Cost_of_shipping_14_5_24 {
```

```
    public static void main(String args[]){
```

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

```
        int wt=sc.nextInt();
```

```
        int d=sc.nextInt();
```

```
        double base=5.0;
```

```
        double costPrKg=2.00;
```

```
        double distPerKm=0.05; //0.5/10=0.05
```

```
        double totalCost=wt*costPrKg + d*distPerKm + base;
```

```
        System.out.print(totalCost);
```

```
    }
```

```
}
```

```
//-----
```

/\*Problem Statement:

Given an array of integers nums and an integer k, your task is to find the length of the longest subarray whose sum equals k.

Example:

Input: nums = [1, -1, 5, -2, 3], k = 3

Output: 4

Explanation: The subarray [1, -1, 5, -2] has the sum 3 and is the longest subarray with this sum.

Example 2:

Input: nums = [-2, -1, 2, 1], k = 1

Output: 2

Explanation: The subarray [-1, 2] has the sum 1 and is the longest subarray with this sum.

Notes:

The subarray should be contiguous.

You need to return the length of the subarray, not the subarray itself.

The array may contain both positive and negative numbers.

\*/

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

```
public class q19_LongestMax_subarraySum_equals_to_k {
```

```
    public static int MaxSubArraySumK(int []nums,int k){
```

```
        int sum=0;
```

```
        int maxLength=0;
```

```
        HashMap<Integer,Integer>preSum=new HashMap<>();
```



```

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

    sum+=nums[i];

    if(sum==k){

        maxLength=i+1;

    }

    int rem=sum-k;

    if(preSum.containsKey(rem)){

        maxLength=Math.max(maxLength,i-preSum.get(rem));

    }

    if(!preSum.containsKey(sum)){

        preSum.put(sum,i);

    }

}

return maxLength;

}

```

```

public static void main(String args[]){

```

```

    Scanner sc=new Scanner(System.in);

```

```

    System.out.println("enter size ");

```

```

    int n=sc.nextInt();

```

```

System.out.println("enter value of k ");

int k=sc.nextInt();

int nums[]=new int[n];

System.out.println("enter array el");

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

nums[i]=sc.nextInt();

}

System.out.println(MaxSubArraySumK(nums,k));

}

}

//-----.

```

```

public class q20_equilibriumPoint {

    public static int equilibriumPoint(long arr[]) {

        // Your code here

        int n=arr.length;

        if(n==1) return (int)arr[0];

        if(n==2) return -1;

        long left[]=new long[n];

        long right[]=new long[n];

        long leftSum=0;

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

```

```

        leftSum+=arr[i];

        left[i]=leftSum;

        // System.out.println(left[i]);
    }

    long rightSum=0;

    for(int j=n-1;j>=0;j--){

        rightSum+=arr[j];

        right[j]=rightSum;

    }

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

        if(right[i]==left[i]){

            return i+1;

        }

    }

    return -1;

}

public static void main(String[] args) {

    long arr[]={1,3,5,2,2};

    System.out.println(equilibriumPoint(arr));

}

}

```

// -----

`/*K Sized Subarray Maximum`

Difficulty: MediumAccuracy: 26.04%Submissions: 319K+Points: 4

Given an array `arr[]` of size `N` and an integer `K`. Find the maximum for each and every contiguous subarray of size `K`.

Example 1:

Input:

`N = 9, K = 3`

`arr[] = 1 2 3 1 4 5 2 3 6`

Output:

`3 3 4 5 5 5 6`

Explanation:

1st contiguous subarray = {1 2 3} Max = 3

2nd contiguous subarray = {2 3 1} Max = 3

3rd contiguous subarray = {3 1 4} Max = 4

4th contiguous subarray = {1 4 5} Max = 5

5th contiguous subarray = {4 5 2} Max = 5

6th contiguous subarray = {5 2 3} Max = 5

7th contiguous subarray = {2 3 6} Max = 6

Example 2:

Input:

`N = 10, K = 4`

`arr[] = 8 5 10 7 9 4 15 12 90 13`

Vaibhav Lanjewar

Output:

10 10 10 15 15 90 90

Explanation:

1st contiguous subarray = {8 5 10 7}, Max = 10

2nd contiguous subarray = {5 10 7 9}, Max = 10

3rd contiguous subarray = {10 7 9 4}, Max = 10

4th contiguous subarray = {7 9 4 15}, Max = 15

5th contiguous subarray = {9 4 15 12},

Max = 15

6th contiguous subarray = {4 15 12 90},

Max = 90

7th contiguous subarray = {15 12 90 13},

Max = 90

<https://www.geeksforgeeks.org/problems/maximum-of-all-subarrays-of-size-k3101/1?page=1&sprint=a663236c31453b969852f9ea22507634&sprint=a663236c31453b969852f9ea22507634&sortBy=submissions>

\*/

```
import java.util.ArrayList;
```

```
import java.util.PriorityQueue;
```

```
public class q21_kSizedSubArrayMximum {
```

```
    static ArrayList <Integer> max_of_subarrays(int arr[], int n, int k)
```

```
{ // only 900/1100 test cases solves o
```

```
    // Your code here
```

```
    ArrayList<Integer>res=new ArrayList<>();
```

```

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

            PriorityQueue<Integer>pq= new PriorityQueue<>((a, b) -> b - a);

            for(int j=i;j<i+k;j++){

                pq.add(arr[j]);

            }

            res.add( pq.peek());

        }

        return res;

    }
}

```

```

public static void main(String[] args) {

    int arr[] = {8, 5, 10, 7, 9, 4, 15, 12, 90, 13};

    int k=3;

    int n = arr.length;

    ArrayList<Integer> result = max_of_subarrays(arr, n, k);

    System.out.println(result.toString());

}

}

//-----

```

```

/* SUM of elements in an array */

```

```

import java.util.*;

public class q22_sum_of_array_elements {

    public static void main(String args[]){

```

```

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int arr[]=new int[n];

int sum=0;

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

    arr[i]=sc.nextInt();

    sum+=arr[i];

}

System.out.println(sum);

}

}

//-----

```

/\* Non-Repeating Element

Find the first non-repeating element in a given array arr of integers and if there is not present any non-repeating element then return 0

Note: The array consists of only positive and negative integers and not zero.

Examples:

Input: arr[] = [-1, 2, -1, 3, 2]

Output: 3

Explanation: -1 and 2 are repeating whereas 3 is the only number occurring once. Hence, the output is 3.

```
*/

import java.util.*;

public class q23_non_repeating_element{

    public static int firstNonRepeating(int[] arr) {

        HashMap<Integer,Integer>mp=new HashMap<>();

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

            mp.put(arr[i],mp.getOrDefault(arr[i],0)+1);

        }

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

            if(mp.get(arr[i])==1){

                return arr[i];

            }

        }

        return -1;

    }

    public static void main(String args[]){
```



```

Scanner sc=new Scanner(System.in);

int size=sc.nextInt();

int arr[]=new int[size];

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

    arr[i]=sc.nextInt();


System.out.print(firstNonRepeating(arr));
}

}

//-----

```

/\* Rotate Array

Given an unsorted array arr[] of size n. Rotate the array to the left (counter-clockwise direction) by d steps, where d is a positive integer.

Note: Consider the array as circular.

Examples:

input: n = 5, d = 2 arr[] = {1,2,3,4,5}

Output: 3 4 5 1 2

Explanation: 1 2 3 4 5 when rotated by 2 elements, it becomes 3 4 5 1 2.

Input: d = 2 , arr[] = {1, 2, 3, 4, 5, 6, 7}

Output: 3 4 5 6 7 1 2

Input: d = 4, arr[] = {3, 4, 5},

Output: 4 5 3

Rotating an array 4 times is effectively same as rotating it  $4 - 3 = 1$  time.

```
*/
```

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

```
public class q24_rotate_array_left{
```

```
    public static void reverse(int arr[],int i,int j){
```

```
        while(i<j){
```

```
            int temp=arr[i];
```

```
            arr[i]=arr[j];
```

```
            arr[j]=temp;
```

```
            i++;
```

```
            j--;
```

```
        }
```

```
    }
```

```
    public static void RightrotateArr(int arr[], int d, int n) {
```

```
        reverse(arr,0,n-1);
```

```
        reverse(arr,0,d-1);
```

```
        reverse(arr,d,n-1);
```

```
}

public static void leftrotateArr(int arr[], int d, int n) {

    reverse(arr,d,n-1);

    reverse(arr,0,d-1);

    reverse(arr,0,n-1);

}
```

```
public static void main(String args[]){

Scanner sc=new Scanner(System.in);

int size=sc.nextInt();

int d=sc.nextInt();

int arr[]=new int[size];

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

    arr[i]=sc.nextInt();

}
```

```
// leftrotateArr(arr,d,size);

RightrotateArr(arr,d,size);

System.out.println("arr after rotation");

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

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

```
}
```

```
}
```

```
}
```

```
//-----
```

```
/*Equal Sum
```

Difficulty: MediumAccuracy: 37.32%Submissions: 29K+Points: 4

Given an array arr. Determine if there exists an element in the array such that the sum of the elements on its left is equal to the sum of the elements on its right. If there are no elements to the left/right, then the sum is considered to be zero.

Example:

Input: arr[] = [1, 2, 3, 3]

Output: true

Explanation: Consider 1-based indexing i = 3, for [1, 2] sum is 3 and for [3] sum is also 3.

Input: arr[] = [1, 5]

Output: false

Explanation: No such index present.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(1)

Constraints:

$1 \leq \text{arr.size()} \leq 10^5$

$1 \leq \text{arr}[i] \leq 10^6$  \*/

```
import java.util.*;

public class q25_equilibriumPoint_gfg_tcs {

    public static String equilibrium(int arr[]) {

        // code here

        int leftSum=0;

        int totSum=0;

        int n=arr.length;

        for(int i:arr){

            totSum+=i;

        }

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

            totSum-=arr[i];

            if(totSum==leftSum){

                return "true";

            }

            leftSum+=arr[i];

        }

        return "false";

    }

    public static void main(String args[]){
```

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

int size=sc.nextInt();

int arr[]=new int[size];

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

System.out.println(equilibrium(arr));

}

}

//-----
```

/\* Reverse a String

Difficulty: BasicAccuracy: 69.49%Submissions: 362K+Points: 1

You are given a string s. You need to reverse the string.

Example 1:

Input:

s = Geeks

Output: skeeG

Example 2:

Input:

s = for

Output: rof

Your Task:

You only need to complete the function `reverseWord()` that takes `s` as parameter and returns the reversed string.

Expected Time Complexity:  $O(|S|)$ .

Expected Auxiliary Space:  $O(1)$ .

Constraints:

$1 \leq |s| \leq 10000$  \*/

```
import java.util.*;

public class q26_reverseString {

    public static String reverseWord(String str)
    {
        char []ch=str.toCharArray();

        StringBuilder sb=new StringBuilder();

        int i=0,j=ch.length-1;
```

```
while(i<j){  
    char temp=ch[i];  
    ch[i]=ch[j];  
    ch[j]=temp;  
    i++;  
    j--;  
}
```

```
for(char c:ch){  
    sb.append(c);  
}  
return sb.toString();
```

```
}
```

```
public static void main(String args[]){  
    Scanner sc=new Scanner(System.in);  
    String str=sc.next();  
    System.out.println(reverseWord(str));
```

```
}
```

```
}
```

```
//-----
```

```
/*
```

Find the smallest and second smallest element

Difficulty: BasicAccuracy: 24.44%Submissions: 105K+Points: 1



Given an array, arr of integers, your task is to return the smallest and second smallest element in the array. If the smallest and second smallest do not exist, return -1.

Examples:

Input: arr[] = [2, 4, 3, 5, 6]

Output: 2 3

Explanation: 2 and 3 are respectively the smallest and second smallest elements in the array.

Input: arr[] = [1, 1, 1]

Output: -1

Explanation: Only element is 1 which is smallest, so there is no second smallest element.

Expected Time Complexity:  $O(n)$

Expected Auxillary Space:  $O(1)$

Constraints:

$1 \leq \text{arr.size} \leq 105$

$1 \leq \text{arr}[i] \leq 105$

\*/

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

```
public class q27_smallestAndSecSmallestElement{
```

```
    public static int[] minAnd2ndMin(int arr[]) {
```

```
        int res[]=new int[2];
```

```
        int small=Integer.MAX_VALUE;
```

```
        int secSmall=Integer.MIN_VALUE;
```

```

for(int i:arr){

    if(i<small){

        secSmall=small;

        small=i;

    }

    if(i<secSmall && i>small)

        secSmall=i;

}

if(small==Integer.MAX_VALUE || secSmall==Integer.MAX_VALUE){

    res[0]=-1;

    res[1]=-1;

}

else{

    res[0]=small;

    res[1]=secSmall;

}

return res;

}

```

```

public static void main(String args[]){

    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 res[]=new int[2];

    res=minAnd2ndMin(arr);

    System.out.println("Smallest element is: "+res[0]);

    System.out.println("Second smallest element is: "+res[1]);

}

}

//-----

```

```

/*

```

## Frequencies of Limited Range Array Elements

Difficulty: EasyAccuracy: 27.64%Submissions: 291K+Points: 2

You are given an array `arr[]` containing positive integers. These integers can be from 1 to `p`, and some numbers may be repeated or absent. Your task is to count the frequency of all numbers that lie in the range 1 to `n`.

Note:

Do not modify the array in-place changes in `arr[]`, such that `arr[i] = frequency(i)` and assume 1-based indexing.

The elements greater than `n` in the array can be ignored when counting.

Examples

Input: `n = 5, arr[] = [2, 3, 2, 3, 5], p = 5`

Output: `[0, 2, 2, 0, 1]`

Explanation: Counting frequencies of each array element We have: 1 occurring 0 times. 2 occurring 2 times. 3 occurring 2 times. 4 occurring 0 times. 5 occurring 1 time, all the modifications done in the same given `arr[]`.

Input: n = 4, arr[] = [3, 3, 3, 3], p = 3

Output: [0, 0, 4, 0]

Explanation: Counting frequencies of each array element We have: 1 occurring 0 times. 2 occurring 0 times. 3 occurring 4 times. 4 occurring 0 times.

Input: n = 2, arr[] = [8, 9], p = 9

Output: [0, 0]

Explanation: Counting frequencies of each array element We have: 1 occurring 0 times. 2 occurring 0 times. Since here P=9, but there are no 9th Index present so can't count the value.

Expected time complexity: O(n)

Expected auxiliary space: O(1)

<https://www.geeksforgeeks.org/problems/frequency-of-array-elements-1587115620/1>

```
*/  
  
import java.util.*;  
  
public class q28_freqArrEl {  
  
    public static void frequencyCount(int arr[], int N, int P) {  
        // do modify in the given array  
        int ans[]=new int[N];  
        Arrays.fill(ans, 0);  
        for(int i=0;i<N;i++){  
            if(arr[i]>=1 && arr[i]<=P && arr[i]<=N){  
                ans[arr[i]-1]++;  
            }  
            else{  
                continue;  
            }  
        }  
    }  
}
```

```

    }

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

        arr[i]=ans[i];

    }

}

public static void main(String args[]){

    Scanner sc=new Scanner(System.in);

    int n=sc.nextInt();

    int p=sc.nextInt();

    int arr[]=new int[n];

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

        arr[i]=sc.nextInt();

    }

    frequencyCount(arr,n,p);

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

}

}

//-----

/*

```

Addition of two square matrices

Difficulty: BasicAccuracy: 63.09%Submissions: 11K+Points: 1

Given two square matrices matrixA and matrixB of size  $n \times n$ . Find the addition of these two matrices.

Note :- You don't have to return anything in this question (Refer YourTask section).

Example 1:

Input: matrixA = {{1, 2}, {3, 4}},

matrixB = {{4, 3}, {2, 1}}

Output: {{5, 5}, {5, 5}}

Example 1:

Input: matrixA = {{2, 2}, {2, 2}},

matrixB = {{3, 3}, {3, 3}}

Output: {{5, 5}, {5, 5}}

Your Task:

You don't need to read or print anything. Your task is to complete the function `Addition()` which takes `matrixA` and `matrixB` as input parameters and adds two matrices. Do the addition without using extra memory. The output matrix must be in `matrixA`.

Expected Time Complexity:  $O(n^2)$

Expected Space Complexity:  $O(1)$

Constraints:

$1 \leq n \leq 100$

```
*/
```

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

```
public class q29_additionOfTwoMatrix {
```

```
    public static void Addition(int[][] matrixA, int[][] matrixB)
```

```
    {
```

```
        int n=matrixA.length;
```

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

```
            for(int j=0;j<n;j++){
```

```
                matrixA[i][j]+=matrixB[i][j];
```

```
            }
```

```
        }
```

```
    }
```

```
    public static void main(String args[]){
```

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

```
        int n=sc.nextInt();
```

```
        int mat1[][]=new int[n][n];
```

```
        int mat2[][]=new int[n][n];
```

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

```
            for(int j=0;j<n;j++){
```

```
                mat1[i][j]=sc.nextInt();
```

```
            }
```

```
        }
```

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

```
            for(int j=0;j<n;j++){
```

```

        mat2[i][j]=sc.nextInt();
    }
}

Addition(mat1,mat2);
for(int i=0;i<n;i++){
    for(int j=0;j<n;j++){
        System.out.print(mat1[i][j]+" ");
    }
}
}

}

```

//-----

/\*

Check if array is sorted

Difficulty: EasyAccuracy: 39.37%Submissions: 210K+Points: 2

Given an array arr[], check whether it is sorted in non-decreasing order. Return true if it is sorted otherwise false.

Examples:

Input: arr[] = [10, 20, 30, 40, 50]

Output: true

Vaibhav Lanjewar



Explanation: The given array is sorted.

Input: arr[] = [90, 80, 100, 70, 40, 30]

Output: false

Explanation: The given array is not sorted.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(1)

Constraints:

$1 \leq \text{arr.size} \leq 106$

$-109 \leq \text{arr}[i] \leq 109$

```
*/  
  
import java.util.*;  
  
public class q30_check_if_arraya_sorted {  
  
    public static boolean arraySortedOrNot(int[] arr, int n) {  
  
        // code here  
  
        for(int i=1;i<n;i++){  
  
            if(arr[i]<arr[i-1]) return false;  
  
        }  
  
        return true;  
  
    }  
  
    public static void main(String args[]){  
  
        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.print(arraySortedOrNot(arr,n));

}

}

//-----

```

/\*

Remove Duplicates from unsorted array

Difficulty: BasicAccuracy: 42.1%Submissions: 24K+Points: 1

Given an array arr of integers which may or may not contain duplicate elements. Your task is to remove duplicate elements.

Examples:

Input: arr[] = [1, 2, 3, 1, 4, 2]

Output: [1, 2, 3, 4]

Explanation: 2 and 1 have more than 1 occurrence.

Input: arr[] = [1, 2, 3, 4]

Output: [1, 2, 3, 4]

Explanation: There is no duplicate element.

Expected Time Complexity:  $O(n)$

Expected Auxiliary Space:  $O(n)$

Constraints:

$1 \leq \text{arr.size()} \leq 10^6$

$1 \leq \text{arr}[i] \leq 10^5$

```

*/

import java.util.*;

public class q31_removeDuplicateFromArray{

    public static ArrayList<Integer> removeDuplicate(int arr[]) {

        // code here

        int n=arr.length;

        ArrayList<Integer>res=new ArrayList<>();

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

            if(!res.contains(arr[i])){

                res.add(arr[i]);

            }

        }

        return res;

    }

    public static void main(String args[]){

        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();

        }

        // Arrays.sort(arr);

        ArrayList<Integer>res=removeDuplicate(arr);
    }
}

```

```
        System.out.println(res);
    }
}
```

```
//-----
```

```
/*
```

### Array Duplicates

Difficulty: EasyAccuracy: 18.95%Submissions: 745K+Points: 2

Given an array arr of size n which contains elements in range from 0 to n-1, you need to find all the elements occurring more than once in the given array. Return the answer in ascending order. If no such element is found, return list containing [-1].

### Examples:

Input: n = 4, arr[] = [0,3,1,2]

Output: -1

Explanation: There is no repeating element in the array. Therefore output is -1.

Input: n = 5, arr[] = [2,3,1,2,3]

Output: 2 3

Explanation: 2 and 3 occur more than once in the given array.

Expected Time Complexity:  $O(n)$ .

Expected Auxiliary Space:  $O(n)$ .

### Constraints:

$1 \leq n \leq 10^5$

0 <= arr[i] <= 105, for each valid i

\*/

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

```
public class q32_RemoveDuplicates {
```

```
    public static ArrayList<Integer> removeDuplicate(int arr[]) {
```

```
        int n=arr.length;
```

```
        ArrayList<Integer>res=new ArrayList<>();
```

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

```
            if(!res.contains(arr[i])){
```

```
                res.add(arr[i]);
```

```
            }
```

```
        }
```

```
        return res;
```

```
    }
```

```
    public static void main(String args[]){
```

```
        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(removeDuplicate(arr));
```

```
}  
}
```

```
//-----
```

```
/*
```

Remove Spaces

Difficulty: BasicAccuracy: 49.21%Submissions: 69K+Points: 1

Given a string, remove spaces from it.

Example 1:

Input:

S = "geeks for geeks"

Output: geeksforgeeks

Explanation: All the spaces have been removed.

Example 2:

Input:

S = " g f g"

Output: gfg

Explanation: All the spaces including the leading ones have been removed.

Your Task:

You don't need to read input or print anything. Your task is to complete the function `modify()` which takes the string `S` as input and returns the resultant string by removing all the white spaces from `S`.

Expected Time Complexity:  $O(|S|)$ .

Expected Auxiliary Space:  $O(1)$ .

Constraints:

$1 \leq |S| \leq 105$  \*/

```
import java.util.*;

public class q33_removeSpaces {

    public static String modify(String S)

    {

        StringBuilder sb=new StringBuilder();

        String ch[]=S.split(" ");

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

            sb.append(ch[i]);

        }

        return sb.toString();

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        String str=sc.nextLine();

        System.out.print(modify(str));

    }

}
```

```
}  
}  
//-----
```

```
/*
```

Remove character

<https://www.geeksforgeeks.org/problems/remove-character3815/1>

Difficulty: BasicAccuracy: 59.6%Submissions: 47K+Points: 1

Given two strings string1 and string2, remove those characters from first string(string1) which are present in second string(string2). Both the strings are different and contain only lowercase characters.

NOTE: Size of first string is always greater than the size of second string(  $|string1| > |string2|$ ).

Example 1:

Input:

string1 = "computer"

string2 = "cat"

Output: "ompuer"

Explanation: After removing characters(c, a, t)

from string1 we get "ompuer".

Example 2:

Input:



```
string1 = "occurrence"
```

```
string2 = "car"
```

Output: "ouene"

Explanation: After removing characters

(c, a, r) from string1 we get "ouene". \*/

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

```
public class q34_removeCharacter {
```

```
    static String removeChars(String string1, String string2){
```

```
        // code here
```

```
        HashSet<Character>set=new HashSet<>();
```

```
        StringBuilder sb=new StringBuilder();
```

```
        for(int i=0;i<string2.length();i++){
```

```
            set.add(string2.charAt(i));
```

```
        }
```

```
        for(int i=0;i<string1.length();i++){
```

```
            if(!set.contains(string1.charAt(i))){
```

```
                sb.append(string1.charAt(i));
```

```
            }
```

```
        }
```

```
        return sb.toString();
```

```
    }
```

```
    public static void main(String args[]){
```

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

```

String str1=sc.nextLine();

String str2=sc.nextLine();

System.out.print(removeChars(str1,str2));

}

}

//-----

```

```

/*

```

Remove all characters other than alphabets

Difficulty: BasicAccuracy: 46.16%Submissions: 16K+Points: 1

Given a string S, remove all the characters other than the alphabets.

Example 1:

Input: S = "\$Gee\*k;s..fo, r'Ge^eks?"

Output: GeeksforGeeks

Explanation: Removed charcters other than alphabets.

Example 2:

Input: S = "{{{}}}> \*& ^%\*")"

Output: -1

Explanation: There are no alphabets.

Your Task:

You don't need to read input or print anything. Your task is to complete the function `removeSpecialCharacter()` which takes string `S` as input parameter and returns the resultant string. Return `"-1"` if no alphabets remain.

Expected Time Complexity:  $O(|s|)$

Expected Auxiliary Space:  $O(|s|)$

Constraints:

$1 \leq |S| \leq 10^5$

```
import java.util.*;

public class q35_removeAllChar {

    public static String removeSpecialCharacter(String s) {

        StringBuilder sb=new StringBuilder();

        for(char i:s.toCharArray()){

            if((i>='a' && i<='z') || (i>='A' && i<='Z')){

                sb.append(i);

            }

        }

        return sb.length()==0?"-1":sb.toString();

    }

}
```

```
public static void main(String args[]){  
  
    Scanner sc=new Scanner(System.in);  
  
    String str=sc.nextLine();  
  
    System.out.print(removeSpecialCharacter(str));  
  
    }  
}  
  
//-----
```

```
/*
```

Reverse a String

You are given a string s. You need to reverse the string.

Example 1:

Input:

s = Geeks

Output: skeeG

Example 2:

Input:

s = for

Output: rof

Your Task:

You only need to complete the function reverseWord() that takes s as parameter and returns the reversed string.

Expected Time Complexity:  $O(|S|)$ .

Expected Auxiliary Space:  $O(1)$ .

Constraints:

$1 \leq |s| \leq 10000$

\*/

```
import java.util.*;

public class q36_reverseString{

    public static String reverseWord(String str){

        char ch[]=str.toCharArray();

        int i=0,j=ch.length-1;

        while(i<j){

            char temp=ch[i];

            ch[i]=ch[j];

            ch[j]=temp;

            i++;

            j--;

        }

        StringBuilder sb=new StringBuilder();

        for(char c:ch){

            sb.append(c);

        }

        return sb.toString();

    }

}
```

```
public static void main(String args[]){  
    Scanner sc=new Scanner(System.in);  
    String str=sc.nextLine();  
    System.out.print(reverseWord(str));  
}  
}
```

```
//-----
```

```
/*
```

Count Alphabets

<https://www.geeksforgeeks.org/problems/count-alphabets3649/1>

Given a string, The task is to count the number of alphabets present in the string.

Example 1:

Input:

S = "adjfh23"

Output: 6

Explanation: only last 2 are not

alphabets.

Example 2:

Input:

S = "n0ji#k\$"

Output: 4

Explanation: #, \$, 0 are not alphabets.

Your Task:

You don't need to read input or print anything. Your task is to complete the function Count() which takes the string S as inputs and returns alphabets count.

Expected Time Complexity:  $O(|S|)$

Expected Auxiliary Space:  $O(1)$

Constraints:

$1 \leq |S| \leq 105$

S contains only upper and lower case alphabets, digits and '#', '!', '\$', '&' only.

\*/

```
import java.util.*;

public class q38_countAlphabets {

    public static int countletter(String S)
    {
        // code here

        int cnt=0;

        for(char i:S.toCharArray()){

            if((i>='a' && i<='z') || i>='A' && i<='Z'){

                cnt++;

            }

        }

    }

}
```

```

        return cnt;
    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        String str=sc.nextLine();

        System.out.print(countletter(str));

    }

}

```

//-----

/\*

### Non Repeating Character

Given a string s consisting of lowercase Latin Letters. Return the first non-repeating character in s. If there is no non-repeating character, return '\$'.

Note : When you return '\$' driver code will output -1.

Examples:

Input:

s = hello

Output: h

Explanation: In the given string, the first character which is non-repeating is h, as it appears first and there is no other 'h' in the string.

Input:

s = zxvczbtxyzvy



Output: c

Explanation: In the given string, 'c' is the character which is non-repeating.

Expected Time Complexity:  $O(n)$ .

Expected Auxiliary Space:  $O(\text{Number of distinct characters})$ .

Note:  $n = |S|$

Constraints:

$1 \leq n \leq 105$

<https://www.geeksforgeeks.org/problems/non-repeating-character-1587115620/1>

\*/

```
import java.util.HashMap;
```

```
import java.util.Scanner;
```

```
public class q39_Non_repeating_character {  
  
    public static char nonrepeatingCharacter(String S)  
    {  
        //Your code here  
  
        HashMap<Character,Integer>mp=new HashMap<>();  
        for(char ch:S.toCharArray()){  
            mp.put(ch,mp.getOrDefault(ch,0)+1);  
        }  
  
        for(char ch:S.toCharArray()){  
            if(mp.get(ch)==1){  
                return ch;  
            }  
        }  
    }  
}
```

```

    }

    return '$';
}

public static void main(String args[]){

    Scanner sc=new Scanner(System.in);

    String str=sc.nextLine();

    System.out.print(nonrepeatingCharacter(str));

}

}

//-----

```

/\* Anagram

Given two strings a and b consisting of lowercase characters. The task is to check whether two given strings are an anagram of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different. For example, act and tac are an anagram of each other. Strings a and b can only contain lower case alphabets.

Note:-

If the strings are anagrams you have to return True or else return False

|s| represents the length of string s.

Example 1:

Input:a = geeksforgeeks, b = forgeeksgeeks

Vaibhav Lanjewar

Output: YES

Explanation: Both the string have same characters with same frequency. So, both are anagrams.

Example 2:

Input:a = allergy, b = allergic

Output: NO

Explanation: Characters in both the strings are not same, so they are not anagrams.

Your Task:

You don't need to read input or print anything. Your task is to complete the function `isAnagram()` which takes the string `a` and string `b` as input parameter and check if the two strings are an anagram of each other. The function returns true if the strings are anagram else it returns false. The driver code will print YES if the function returns true, else print NO.

Expected Time Complexity: $O(|a| + |b|)$ .

Expected Auxiliary Space:  $O(\text{Number of distinct characters})$ .

Constraints:

$1 \leq |a|, |b| \leq 10^5$

\*/

```
import java.util.Arrays;
```

```
import java.util.Scanner;
```

```
public class q40_AnaGram {
```

```
    public static boolean isAnagram(String a, String b) {
```

```

// Your code here

if(a.length()>b.length() || a.length()<b.length()) return false;

char[]s1=a.toCharArray();

char[]s2=b.toCharArray();

Arrays.sort(s1);

Arrays.sort(s2);

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

    if(s1[i]!=s2[i]){

        return false;

    }

}

return true;

}

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

String str1=sc.nextLine();

String str2=sc.nextLine();

System.out.print(isAnagram(str1,str2));

}

}

//-----

```

/\*Sort a String

Difficulty: BasicAccuracy: 64.43%Submissions: 30K+Points: 1

Given a string consisting of lowercase letters, arrange all its letters in ascending order.

Example 1:

Input:

S = "edcab"

Output: "abcde"

Explanation: characters are in ascending order in "abcde".

Example 2:

Input:

S = "xzy"

Output: "xyz"

Explanation: characters are in ascending order in "xyz".

Your Task:

You don't need to read input or print anything. Your task is to complete the function `sort()` which takes the string as inputs and returns the modified string.

Expected Time Complexity:  $O(|S| * \log |S|)$

Expected Auxiliary Space:  $O(1)$

Constraints:

$1 \leq |S| \leq 10^5$

S contains only lowercase alphabets. \*/

// <https://www.geeksforgeeks.org/problems/sort-a-string2943/1>

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

Vaibhav Lanjewar

```

public class q41_SortString_char {

    public static String sort(String s){

        char ch[]=s.toCharArray();

        Arrays.sort(ch);

        String b = new String(ch);

        return b;

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        String str=sc.nextLine();

        System.out.print(sort(str));

    }

}

//-----

```

/\* Implement strstr

Difficulty: BasicAccuracy: 46.9%Submissions: 176K+Points: 1

Your task is to implement the function strstr. The function takes two strings as arguments (s,x) and locates the occurrence of the string x in the string s. The function returns an integer denoting the first occurrence of the string x in s (0 based indexing).

Note: You are not allowed to use inbuilt function.

Example 1:

Input:

`s = GeeksForGeeks, x = Fr`

Output: -1

Explanation: Fr is not present in the string GeeksForGeeks as substring.

Example 2:

Input:

`s = GeeksForGeeks, x = For`

Output: 5

Explanation: For is present as substring in GeeksForGeeks from index 5 (0 based indexing).

Your Task:

You don't have to take any input. Just complete the `strstr()` function which takes two strings `str`, `target` as an input parameter. The function returns -1 if no match is found else it returns an integer denoting the first occurrence of the `x` in the string `s`.

Expected Time Complexity:  $O(|s| * |x|)$

Expected Auxiliary Space:  $O(1)$

Note : Try to solve the question in constant space complexity.

Constraints:

$1 \leq |s|, |x| \leq 100$

[//https://www.geeksforgeeks.org/problems/implement-strstr/1](https://www.geeksforgeeks.org/problems/implement-strstr/1)

\*/

```
import java.util.*;

public class q42_CHeck_if_StringisSubStringOfAnother {

    public static int strstr(String s, String x)

    {

        if(s.contains(x)){

            return s.indexOf(x);

        }

        else{

            return -1;

        }

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        String str=sc.nextLine();

        String x=sc.nextLine();

        System.out.print(strstr(str,x));

    }
```



```
}
```

```
//-----
```

```
/*Reverse Words
```

Difficulty: EasyAccuracy: 56.08%Submissions: 320K+Points: 2

Given a String S, reverse the string without reversing its individual words. Words are separated by dots.

<https://www.geeksforgeeks.org/problems/reverse-words-in-a-given-string5459/1>

Example 1:

Input:

S = i.like.this.program.very.much

Output: much.very.program.this.like.i

Explanation: After reversing the whole string(not individual words), the input string becomes

much.very.program.this.like.i

Example 2:

Input:

S = pqr.mno

Output: mno.pqr

Explanation: After reversing the whole string , the input string becomes  
mno.pqr

Your Task:

You dont need to read input or print anything. Complete the function `reverseWords()` which takes string `S` as input parameter and returns a string containing the words in reversed order. Each word in the returning string should also be separated by ' '.

Expected Time Complexity:  $O(|S|)$

Expected Auxiliary Space:  $O(|S|)$

Constraints:

$1 \leq |S| \leq 105$

```
*/  
  
import java.util.*;  
  
public class q43_ReverseWordsInString {  
  
    public static String reverseWords(String S){  
  
        S+=" ";  
  
        String [] str=S.split("\\. ");  
  
        // reverse string  
  
        int i=0,j=str.length-1;  
  
        while(i<j){  
  
            String temp=str[i];  
  
            str[i]=str[j];  
  
            str[j]=temp;  
  

```

```

        i++;

        j--;

    }

// add . to string

    StringBuilder sb=new StringBuilder();

    int n=str.length;

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

        sb.append(str[k]);

        if(k<n-1){

            sb.append(".");

        }

    }

    return sb.toString();

}

public static void main(String args[]){

    Scanner sc=new Scanner(System.in);

    String str=sc.nextLine();

    System.out.print(reverseWords(str));

}

}

//-----

/* Missing in Array

```

Given an array `arr` of size  $n-1$  that contains distinct integers in the range of 1 to  $n$  (inclusive), find the missing element. The array is a permutation of size  $n$  with one element missing. Return the missing element.

Examples:

Input:  $n = 5$ , `arr[] = [1,2,3,5]`

Output: 4

Explanation : All the numbers from 1 to 5 are present except 4.

Input:  $n = 2$ , `arr[] = [1]`

Output: 2

Explanation : All the numbers from 1 to 2 are present except 2.

Expected Time Complexity:  $O(n)$

Expected Auxiliary Space:  $O(1)$

Constraints:

$1 \leq n \leq 10^5$

$1 \leq \text{arr}[i] \leq n$

<https://www.geeksforgeeks.org/problems/missing-number-in-array1416/1>

\*/

import java.util.\*;

public class q44FindMissingNumInArray {

public static int missingNumber(int n, int arr[]) {

// Your Code Here

int sum=0;

```

        for(int i:arr){

            sum+=i;

        }

        int tot=(n*(n+1))/2;

        return tot-sum;

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        int size=sc.nextInt();

        int arr[]=new int[size];

        for(int i=0;i<size-1;i++){

            arr[i]=sc.nextInt();

        }

        System.out.print(missingNumber(size,arr));

    }

}

//-----

```

/\* Kadane's Algorithm

Difficulty: MediumAccuracy: 36.28%Submissions: 969K+Points: 4

Given an integer array arr[]. Find the contiguous sub-array(containing at least one number) that has the maximum sum and return its sum.

//<https://www.geeksforgeeks.org/problems/kadanes-algorithm-1587115620/1>

Examples:

Input: arr[] = [1, 2, 3, -2, 5]

Vaibhav Lanjewar

Output: 9

Explanation: Max subarray sum is 9 of elements (1, 2, 3, -2, 5) which is a contiguous subarray.

Input: arr[] = [-1, -2, -3, -4]

Output: -1

Explanation: Max subarray sum is -1 of element (-1)

Input: arr[] = [5, 4, 7]

Output: 16

Explanation: Max subarray sum is 16 of element (5, 4, 7)

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(1)

Constraints:

$1 \leq \text{arr.size()} \leq 105$

$-107 \leq \text{arr}[i] \leq 107$

\*/

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

```
public class q45KadanesAlgorithm_MaximumSubArraySum {
```

```
    public static int maxSubarraySum(int[] arr) {
```

```
        // Your code here
```

```
        int sum=0;
```

```
        int maxSum=Integer.MIN_VALUE;
```

```
        for(int i:arr){
```

```
            if(sum<0){
```

```
                sum=0;
```

```

    }

    sum+=i;

    maxSum=Math.max(sum,maxSum);

}

return maxSum==Integer.MIN_VALUE?-1 :maxSum;

}

```

```

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

int size=sc.nextInt();

int arr[]=new int[size];

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

    arr[i]=sc.nextInt();

}

System.out.print(maxSubarraySum(arr));

}

```

```

}

```

```

//-----

```

```

/* Nth Natural Number

```

Difficulty: MediumAccuracy: 29.99%Submissions: 56K+Points: 4

Given a positive integer n. You have to find nth natural number after removing all the numbers containing the digit 9.

Examples :

Vaibhav Lanjewar

Input: n = 8

Output: 8

Explanation: After removing natural numbers which contains digit 9, first 8 numbers are 1,2,3,4,5,6,7,8 and 8th number is 8.

Input: n = 9

Output: 10

Explanation: After removing natural numbers which contains digit 9, first 9 numbers are 1,2,3,4,5,6,7,8,10 and 9th number is 10.

Expected Time Complexity:  $O(\log n)$

Expected Auxiliary Space:  $O(1)$

Constraints:

$1 \leq n \leq 10^9$

<https://www.geeksforgeeks.org/problems/nth-natural-number/1>

\*/

import java.util.\*;

public class q46\_FindNthNaturalNumber\_AfterRemovingNine {

public static long findNth(long n){

long res=0;

long place=1;

while(n>0){

res+=(n%9)\*place;

n/=9;

place\*=10;



```

    }

    return res;
}

public static void main(String args[]){

    Scanner sc=new Scanner(System.in);

    long n=sc.nextLong();

    System.out.println(findNth(n));

}

}

//-----

```

/\* Problem Statement: Given an array of integers where every element appears an even number of times except one element which appears an odd number of times, write a program to find that odd number of times occurring element.

Sample Test Case 1:

n = 7

arr = [1, 1, 2, 2, 2, 3, 3]

Output: 2

Sample Test Case 2:

n = 5

arr = [2, 2, 3, 1, 1]

Output: 3 \*/

```
//compile and run

//javac q47_FindOddTimeRepeatedElement.java

//java q47_FindOddTimeRepeatedElement


import java.util.*;

public class q47_FindOddTimeRepeatedElement{


    public static int findOddOccEle(int arr[],int n){

        HashMap<Integer,Integer>mp=new HashMap<>();

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

            mp.put(arr[i],mp.getOrDefault(arr[i],0)+1);

        }


        for(int i:arr){

            if(mp.get(i)%2==1){

                return i;

            }

        }

        return -1;

    }


    public static void main(String args[]){

        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.print(findOddOccEle(arr,n));

}

}

//-----

/*

```

TCS NQT Coding Question 2023 – September Day 1 – Slot 1

Problem Statement –

// VERY IMP

Joseph is learning digital logic subject which will be for his next semester. He usually tries to solve unit assignment problems before the lecture. Today he got one tricky question. The problem statement is “A positive integer has been given as an input. Convert decimal value to binary representation. Toggle all bits of it after the most significant bit including the most significant bit. Print the positive integer value after toggling all bits”.

Constrains-

$1 \leq N \leq 100$

Example 1:

Input :

10 -> Integer

Output :

5 -> result- Integer

Explanation:

Binary representation of 10 is 1010. After toggling the bits(1010), will get 0101 which represents "5". Hence output will print "5".

```
*/
```

```
import java.util.*;

public class q48_toggleBits{

    public static int toggle(int n){

        double lg=Math.log(n)/Math.log(2);

        int k=(1<<(int)lg+1)-1;

        return n^k;

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        System.out.print(toggle(n));

    }

}
```

```
//-----
```

### /\*TCS NQT Coding Question Day 1 Slot 2 – Question 1

Jack is always excited about Sunday. It is favourite day, when he gets to play all day. And goes to cycling with his friends.

So every time when the months starts he counts the number of sundays he will get to enjoy. Considering the month can start with any day, be it Sunday, Monday.... Or so on.

Count the number of Sunday jack will get within n number of days.

Example 1:

Input

mon-> input String denoting the start of the month.

13 -> input integer denoting the number of days from the start of the month.

Output :

2 -> number of days within 13 days.

Explanation:

The month start with mon(Monday). So the upcoming Sunday will arrive in next 6 days. And then next Sunday in next 7 days and so on.

Now total number of days are 13. It means 6 days to first Sunday and then remaining 7 days will end up in another Sunday. Total 2 Sundays may fall within 13 days.

```

*/

import java.util.*;

public class q49_countNoOfSundayWithinNdays {

    public static void main(String args[]){

    }

}

//-----

public class q50_MaximumConsecutiveOnes {

    public static void main(String args[]){

        int arr[]={0,1,1,0,1,1,1,1,0};

        int m=Integer.MIN_VALUE;

        int cnt=0;

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

            if(arr[i]!=0 ){

                cnt++;

            }

            else{

                m=Math.max(m,cnt);

                cnt=0;

            }

        }

        System.out.println(m);

    }

}

```

```
//-----
```

```
import java.util.Arrays;
```

```
public class q51_IsPrime_SeiveOfEratoSthenes {
```

```
    public static void main(String args[]){
```

```
        int n=20;
```

```
        boolean isPrime[]=Checkprime(n);
```

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

```
            if(isPrime[i]==true){
```

```
                System.out.print(i+" ");
```

```
            }
```

```
        }
```

```
    }
```

```
    public static boolean[] Checkprime(int n){
```

```
        boolean []isPrime=new boolean[n+1];
```

```
        Arrays.fill(isPrime,true);
```

```
        isPrime[0]=false;
```

```
        isPrime[1]=false;
```

```
        for(int i=2;i*i<=n;i++){
```

```
            for(int j=2*i;j<=n;j+=i){
```

```
                isPrime[j]=false;
```

```
            }
```

```
        }
```

```
        return isPrime;
```

```

    }

}

//-----

public class q52_GCDofTwoNumbers {

    public static void main(String args[]){

        int a=55;

        int b=11;

        System.out.println(gcd(a,b));

    }

    public static int gcd(int a,int b){

        if(b==0) return a;

        return gcd(b,a%b);

    }

}

//-----

```

/\*Sum of AP series

Difficulty: BasicAccuracy: 31.98%Submissions: 21K+Points: 1

A series with same common difference is known as arithmetic series. The first term of series is 'a' and common difference is d. The series looks like a, a + d, a + 2d, a + 3d, . . . Find the sum of series upto nth term.



Example 1:

Input:  $n = 5$ ,  $a = 1$ ,  $d = 3$

Output: 35

Explanation: Series upto 5th term is

1 4 7 10 13, so sum will be 35.

Example 2:

Input:  $n = 3$ ,  $a = 1$ ,  $d = 2$

Output: 9

Example: Series upto 3rd term is

1 3 5, so sum will be 9.

Your Task:

You don't need to read or print anything. Your task is to complete the function `sum_of_ap()` which takes  $n$ ,  $a$  and  $d$  as input parameter and returns the sum of the series.

Expected Time Complexity:  $O(1)$

Expected Space Complexity:  $O(1)$

Constranits:

$1 \leq n, a, d \leq 10000$

<https://www.geeksforgeeks.org/problems/sum-of-ap-series4512/1>

Vaibhav Lanjewar

```

*/

import java.util.*;

public class q54Numerical_SumOfApSeries {

    public static long sum_of_ap(long n, long a, long d) {

        long sum = 0;

        return n * (2 * a + (n - 1) * d) / 2;

    }

```

```

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        long n = sc.nextLong(); // n

        long a = sc.nextLong(); // first term

        long d = sc.nextLong(); // common diff

        System.out.print(sum_of_ap(n, a, d));

    }

}

//-----

```

```

import java.util.*;

public class q55_factorsOfNumber {

    static int isPrime(int n){

```

```

        int[] isPrim =new int[n+1];

        Arrays.fill(isPrim,1);

        isPrim[0]=0;

        isPrim[1]=0;

        for(int i=2;i*i<=n;i++){

            for(int j=2*i;j<=n;j+=i){

                isPrim[j]=0;

            }

        }

        int cnt=0;

        for(int i:isPrim){

            if(i==1){

                cnt++;

            }

        }

        return cnt;

    }

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        System.out.print(isPrime(n));

    }

}

//-----

import java.util.Scanner;

```

```

public class q56_Replace0sby5 {

    public static int convertFive(int n){

        //add code here.

        String num=Integer.toString(n);

        StringBuilder sb=new StringBuilder();

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

            char ch=num.charAt(i);

            if(ch=='0'){

                sb.append('5');

            }else{

                sb.append(ch);

            }

        }

        return Integer.parseInt(sb.toString());

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        System.out.println(convertFive(n));

    }

}

//-----

```

/\*An integer number in base 10 which is divisible by the sum of its digits is said to be a Harshad Number. An n-Harshad number is an integer number divisible by the sum of its digit in base n.

Below are the first few Harshad Numbers represented in base 10:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 20.....

Given a number in base 10, our task is to check if it is a Harshad Number or not.

Examples :

Input: 3

Output: 3 is a Harshad Number

Input: 18

Output: 18 is a Harshad Number

Input: 15

Output: 15 is not a Harshad Number

[//https://www.geeksforgeeks.org/harshad-or-niven-number/](https://www.geeksforgeeks.org/harshad-or-niven-number/)

\*/

```
import java.util.Scanner;
```

```
public class q57HarshadNumber {  
  
    public static boolean checkHarshad(int n){  
  
        int sum=0;  
  
        for(int temp=n;temp>0;temp/=10){  
  
            sum+=temp%10;  
  
        }  
  
        return (n%sum==0);  
  
    }  
  
}
```

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

```
        System.out.println(checkHarshad(n));
    }
}

//Vaibhav Lanjewar

//-----
```

```
import java.util.Scanner;

public class q58IsLeapYear {

    public static int isLeap(int N){

        //code here

        if(N%100 !=0 && N%4==0 || N%400==0){

            return 1;

        }

        else{

            return 0;

        }

    }

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        int yr=sc.nextInt();

        System.out.println(isLeap(yr));

    }

}

//-----
```

<https://www.geeksforgeeks.org/problems/add-two-fractions/1>

```
import java.util.Scanner;

public class q59AddFracions {

    public static int findGcd(int a, int b){

        if(b==0)

            return a;

        return findGcd(b, a%b);

    }

    public static void addFraction(int num1, int den1, int num2, int den2)

    {

        // Your code here


        int d = den1*den2;

        int n = num1*den2 + num2* den1;


        int gcd = findGcd(n,d);

        n /= gcd;

        d /= gcd;


        System.out.println(n+"/"+d);

    }

    public static void main(String []args){

        Scanner sc=new Scanner(System.in);

        int num1=sc.nextInt();
```

```
        int num2=sc.nextInt();

        int den1=sc.nextInt();

        int den2=sc.nextInt();

    }

}

//-----

/*Sum Of Digits
```

Difficulty: BasicAccuracy: 67.08%Submissions: 42K+Points: 1

Given a number, N. Find the sum of all the digits of N

Example 1:

Input:

N = 12

Output:

3

Explanation:

Sum of 12's digits:

$1 + 2 = 3$

Example 2:

Input:

N = 23

Output

5



Explanation:

Sum of 23's digits:

$$2 + 3 = 5$$

Your Task:

You don't need to read input or print anything. Your task is to complete the function `sumOfDigits()` which takes an integer `N` as input parameters and returns an integer, total sum of digits of `N`.

Expected Time Complexity:  $O(\log_{10} N)$

Expected Space Complexity:  $O(1)$  \*/

```
import java.util.Scanner;
```

```
public class q60SumOfDigits {  
  
    public static int sumOfDigits(int N) {  
  
        int sum=0;  
  
        while(N>0){  
  
            sum+=N%10;  
  
            N=N/10;  
  
        }  
  
        return sum;  
  
    }  
  
    public static void main(String args[]){  
  
        Scanner sc=new Scanner(System.in);  
  
        int n=sc.nextInt();  
  
        System.out.println(sumOfDigits(n));  
  
    }  
}
```

```

}

//-----

import java.util.ArrayList;

public class q61SubsetOfString {

    public static void subset(String str,String ans,int i){

        if(i==str.length()){

            System.out.println(ans);

            return ;

        }

        // choice yes

        subset(str,ans+str.charAt(i),i+1);

        // choise is no

        subset(str,ans,i+1);

    }

    public static void SubsetList(String str,ArrayList<String> ans,String up,int i){

        if(i==str.length()){

            ans.add(up);

            return ;

        }

        // choice yes

```

```

        subset(str,up+str.charAt(i),i+1);

        // chose is no
        subset(str,up,i+1);

    }

    public static void main(String args[]){

        String str="abc";

        // subset(str, "", 0);

        ArrayList<String>ans=new ArrayList<>();

        SubsetList(str,ans,"",0);

        System.out.println(ans);

    }

}

//-----

public class q62MergeSort {

    public static void mergeSort(int[] arr, int si, int ei) {

        if (si >=ei) {

            return;

        }

        int mid = si +(ei-si)/2;

        // left part

        mergeSort(arr, si, mid);

```

```

// right part
mergeSort(arr, mid + 1, ei);

// merge
merge(arr, si, mid, ei);
}

public static void merge(int[] arr, int si, int mid, int ei) {
    int temp[] = new int[ei - si + 1];

    int i = si;

    int j = mid + 1;

    int k = 0; // to iterate the temp array

    while(i <= mid && j <= ei){
        if(arr[i] < arr[j]){
            temp[k++] = arr[i++];
        }
        else{
            temp[k++] = arr[j++];
        }
    }

    while(i <= mid){
        temp[k++] = arr[i++];
    }

    while(j <= ei){
        temp[k++] = arr[j++];
    }
}

```

```

    }

    // copy the element from temp to the arr
    for(k=0,i=si;k<temp.length;k++,i++){
        arr[i]=temp[k];
    }
}

public static void print(int arr[]) {
    for (int i : arr) {
        System.out.print(i + " ");
    }
    System.out.println();
}

public static void main(String args[]) {
    int arr[] = { 6, 3, 95, 2, 8 };
    print(arr);
    mergeSort(arr, 0, arr.length - 1);
    print(arr);
}
}

//-----

```

## `/*884. Uncommon Words from Two Sentences`

Solved

Easy

Topics

Companies

A sentence is a string of single-space separated words where each word consists only of lowercase letters.

A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences `s1` and `s2`, return a list of all the uncommon words. You may return the answer in any order.

Example 1:

Input: `s1 = "this apple is sweet"`, `s2 = "this apple is sour"`

Output: `["sweet", "sour"]`

Explanation:

The word "sweet" appears only in `s1`, while the word "sour" appears only in `s2`.

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

<https://leetcode.com/problems/uncommon-words-from-two-sentences/description/?envType=daily-question&envId=2024-09-17>

\*/

```
import java.util.ArrayList;
```

```
import java.util.HashMap;
```

```
public class q63UncommonWordsFrom2Sent {  
  
    public static String[] uncommonFromSentences(String s1, String s2) {  
  
        String []str1=s1.split(" ");  
  
        String []str2=s2.split(" ");  
  
        HashMap<String,Integer>mp=new HashMap<>();  
  
        for(String s:str1){  
            mp.put(s,mp.getOrDefault(s,0)+1);  
        }  
    }  
}
```

```

        for(String s:str2){

            mp.put(s,mp.getOrDefault(s,0)+1);

        }

        ArrayList<String>res=new ArrayList<>();

        for(String key : mp.keySet()){

            if(mp.get(key)==1){

                res.add(key);

            }

        }

        return res.toArray(new String[0]);

    }

    public static void main(String args[]){

        String s1 = "this apple is sweet", s2 = "this apple is sour";

        String arr[]=uncommonFromSentences(s1,s2);

        for(String i:arr){

            System.out.println(i);

        }

    }

}

//-----

public class q64EvenOddString {

```



```
public static String oddEven(String s) {  
  
    // code here  
  
    // HashMap<Character,Integer>mp=new HashMap<>();  
  
    // int arr[]=new int[26];  
  
    char ch[]=s.toCharArray();  
  
    int odd=0;  
  
    int even=0;  
  
    for(char c:ch){  
  
        int a=c-'a' +1;  
  
        if(a%2!=0){  
  
            odd++;  
  
        }  
  
        else{  
  
            even++;  
  
        }  
    }  
  
    if(odd%2==0){  
  
        odd=0;  
  
    }  
  
    if(even%2!=0){  
  
        even=0;  
  
    }  
  
    return (odd+even)%2==0?"EVEN":"ODD";  
}  
  
public static void main(String[] args) {
```

```

        String s= "abbbcc";

        System.out.println(oddEven(s));

    }

}

//-----

```

/\*Binary number to decimal number

Difficulty: BasicAccuracy: 51.5%Submissions: 63K+Points: 1

Given a Binary Number B, find its decimal equivalent.

Example 1:

Input: B = 10001000

Output: 136

Example 2:

Input: B = 101100

Output: 44 \*/

```

public class q65Binary2Dec{

    public static void main(String[] args) {

        String str="10001000";

        int res=0;

        int n=str.length();

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

```

```

        int no = str.charAt(i) - '0';

        int pow=(int)Math.pow(2,n-i-1);

        res+=pow*no;

    }

    System.out.println(res);

}

```

//-----

```

public class q66Decimal2Binary {

    public static void main(String args[]){

        int N=7;

        int res=0;

        int fact=1;

        while(N>0){

            res+=N%2*fact;

            fact*=10;

            N/=2;

        }

        System.out.println(res);

    }

}

```

//-----

`/*acing the sun`

Difficulty: EasyAccuracy: 45.54%Submissions: 57K+Points: 2

Given an array height representing the heights of buildings. You have to count the buildings that will see the sunrise (Assume the sun rises on the side of the array starting point).

Note: The height of the building should be strictly greater than the height of the buildings left in order to see the sun.

Input: height = [7, 4, 8, 2, 9]

Output: 3

Explanation: As 7 is the first element, it can see the sunrise. 4 can't see the sunrise as 7 is hiding it. 8 can see. 2 can't see the sunrise. 9 also can see

the sunrise.

Input: height = [2, 3, 4, 5]

Output: 4

Explanation: As 2 is the first element, it can see the sunrise. 3 can see the sunrise as 2 is not hiding it. Same for 4 and 5, they also can see the sunrise.

<https://www.geeksforgeeks.org/problems/facing-the-sun2126/1>

`*/`

```
public class q67FacingSum {  
  
    public static int countBuildings(int[] height) {  
  
        // code here  
  
        int see=1;  
  
        int sun=height[0];  
  
        for(int i:height ){  
  
            if(sun<i){  
  
                sun=i;  
  
                see++;  
  
            }  
  
        }  
  
    }  
}
```

```
        return see;
    }

    public static void main(String[] args) {

        int []heights={7, 4, 8, 2, 9};

        System.out.println(countBuildings(heights));

    }
}

//-----

//https://github.com/Vaibhavlansjewar/DsaJava/tree/main/TCS-NQT

// Vaibhav Lankjewar
```

