RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23331

DESIGN AND ANALYSIS OF ALGORITHM LAB

Laboratory Observation Note Book

Name : Afra Fathima

Year / Branch / Section : 2nd Year/ AIML / A

Register No. : 231501007

Semester : 3rd Semester

Academic Year : 2024-2025

S. No.	Date	Title	Page No.	Teacher's Signature / Remarks
Basic C Programming				
1.1	/ /24	Basic C Programming - Practice		
Finding Time Complexity of Algorithms				
2.1	/ /24	Problem 1: Finding Complexity using Counter Method		
2.2	/ /24	Problem 2: Finding Complexity using Counter Method		
2.3	/ /24	Problem 3: Finding Complexity using Counter Method		
2.4	/ /24	Problem 4: Finding Complexity using Counter Method		
2.5	/ /24	Problem 5: Finding Complexity using Counter Method		
Divide and Conquer				
3.1	/ /24	1-Number of Zeros in a Given Array		
3.2	/ /24	2-Majority Element		
3.3	/ /24	3-Finding Floor Value		
3.4	/ /24	4-Two Elements sum to x		
3.5	/ /24	6-Implementation of Quick Sort		
Greedy Algorithms				
4.1	/ /24	1-G-Coin Problem		
4.2	/ /24	2-G-Cookies Problem		
4.3	/ /24	3-G-Burger Problem		
4.4	/ /24	4-G-Array Sum max problem		
4.5	/ /24	5-G-Product of Array elements-Minimum		
Dynamic Programming				
5.1	/ /24	1-DP-Playing with Numbers		
5.2	/ /24	2-DP-Playing with chessboard		
5.3	/ /24	3-DP-Longest Common Subsequence		
5.4	/ /24	4-DP-Longest non-decreasing Subsequence		
Competitive Programming				
6.1	/ /24	1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity		
6.2	/ /24	2-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity		
6.3	/ /24	3-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity		

WEEK 01

BASIC C PROGRAMMING

1)Given two numbers, write a C program to swap the given numbers.

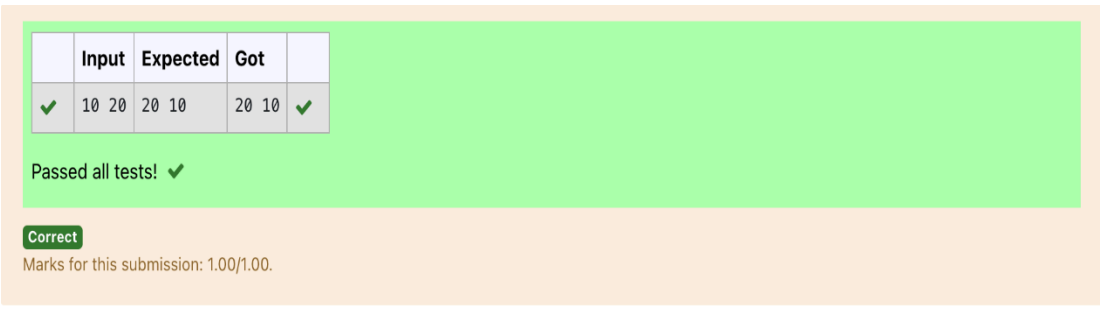
For example:

Input	Result
10 20	20 10

CODE:

```
#include<stdio.h>
int main()
{
   int a,b,temp;
   scanf("%d %d",&a,&b);
   temp=a;
   a=b;
   b=temp;
   printf("%d %d",a,b);
}
```

OUTPUT:



2) Write a C program to find the eligibility of admission for a professional course based on the following criteria:
Marks in Maths >= 65
Marks in Physics >= 55
Marks in Chemistry >= 50
Or
Total in all three subjects >= 180

Sample Test Cases
Test Case 1:
Input:
70 60 80
Output:
The candidate is eligible

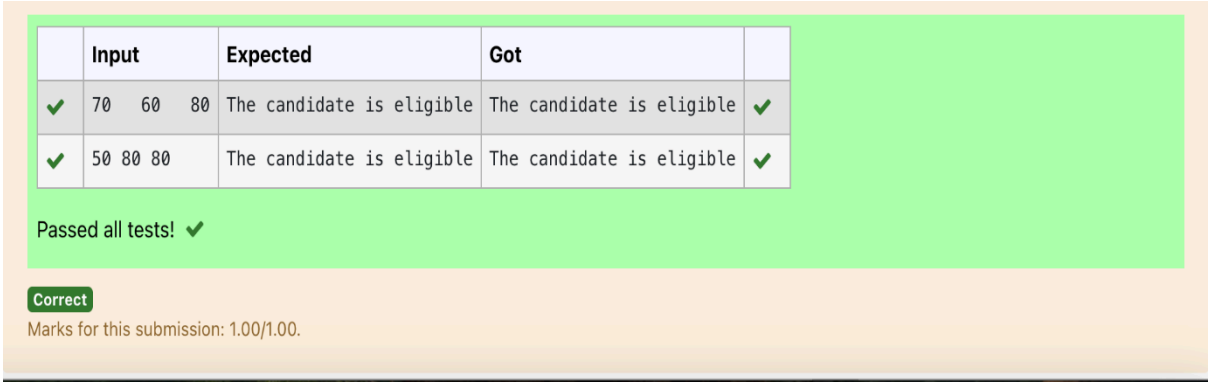
Test Case 2
Input:
50 60 40
Output:
The candidate is not eligible

CODE:

```
#include<stdio.h>
int main()
{
   int m,p,c,t;
   scanf("%d %d %d",&m,&p,&c);
   t=m+p+c;
   if(t>=180 ||(m>=65 && p>=55 && c>=50))
   {
```

```
        printf("The candidate is eligible");
    }
    else
    {
        printf("The candidate is not eligible");
    }
}
```

OUTPUT:



	Input	Expected	Got	
✔	70 60 80	The candidate is eligible	The candidate is eligible	✔
✔	50 80 80	The candidate is eligible	The candidate is eligible	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3) Malini goes to BestSave hyper market to buy grocery items. BestSave hyper market provides 10% discount on the bill amount B when ever the bill amount B is more than Rs.2000.
The bill amount B is passed as the input to the program. The program must print the final amount A payable by Malini.

Input_Format:
The first line denotes the value of B.

Output_Format:
The first line contains the value of the final payable amount A.

Example1:
Input:
1900
Output:
1900

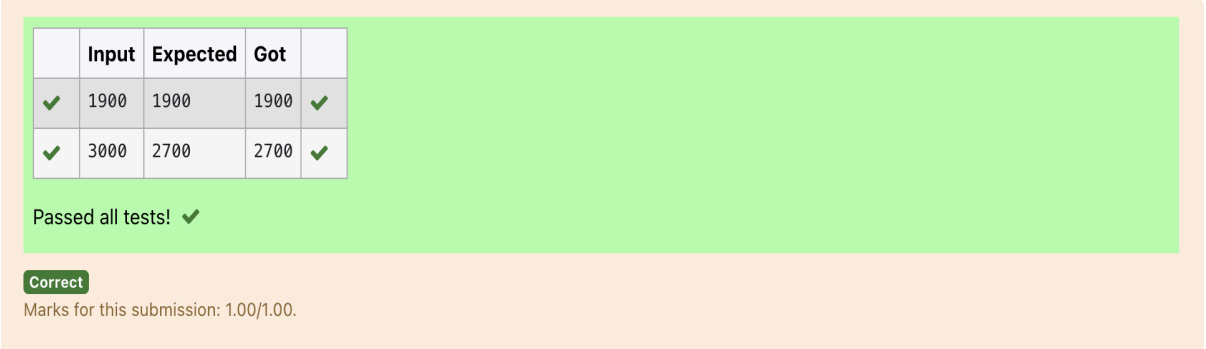
Example2:
Input:
3000
Output:
2700

CODE:

```
#include<stdio.h>
int main()
{
   int b,d;
   scanf("%d",&b);
   if(b>2000)
   {
      d=b*0.1;
      b=b-d;
      printf("%d",b);
   }
   else
   {
      printf("%d",b);
```

```
    }
}
```

OUTPUT:



	Input	Expected	Got	
✔	1900	1900	1900	✔
✔	3000	2700	2700	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4) Baba is very kind to beggars and every day Baba donates half of the amount he has when ever a beggar requests him. The money M left in Baba's hand is passed as the input and the number of beggars B who received the alms are passed as the input. The program must print the money Baba had in the beginning of the day.

Input_Format:

The first line denotes the value of A.
The second line denotes the value of B.

Output_Format:

The first line denotes the value of money with Baba in the beginning of the day.

Example:
Input:

100
2

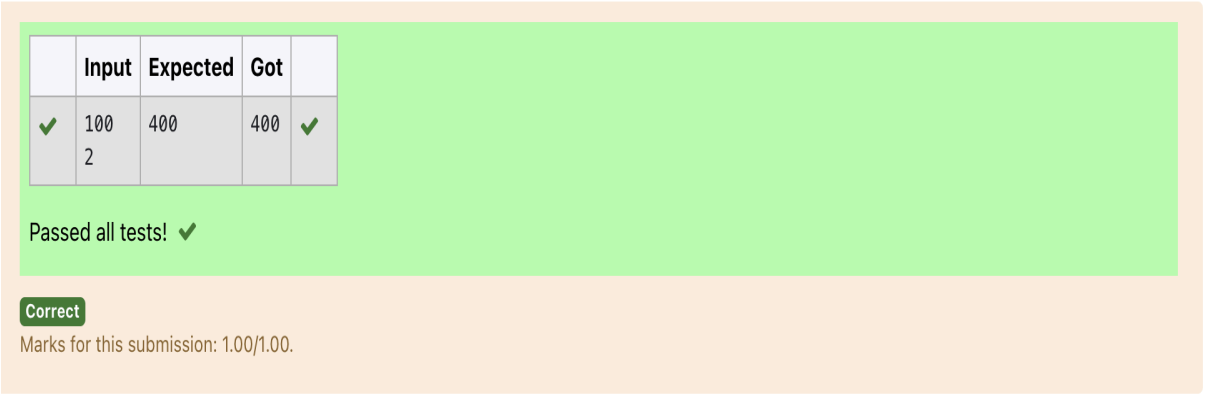
Output:
400

Explanation:
Baba donated to two beggars. So when he encountered second beggar he had 100*2 = Rs.200 and when he encountered 1st he had 200*2 = Rs.400.

CODE:

```
#include<stdio.h>
int main()
{
   int i,m,b;
   scanf("%d %d",&m,&b);
   for(i=0;i<b;i++)
   {
      m=m*b;
   }
   printf("%d",m);
}
```

OUTPUT:



	Input	Expected	Got	
✔	100 2	400	400	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5) The CEO of company ABC Inc wanted to encourage the employees coming on time to the office. So he announced that for every consecutive day an employee comes on time in a week (starting from Monday to Saturday), he will be awarded Rs.200 more than the previous day as "Punctuality Incentive". The incentive I for the starting day (ie on Monday) is passed as the input to the program. The number of days N an employee came on time consecutively starting from Monday is also passed as the input. The program must calculate and print the "Punctuality Incentive" P of the employee.

Input_Format:
The first line denotes the value of I.
The second line denotes the value of N.

Output_Format:
The first line denotes the value of P.

Example:
Input:
500
3

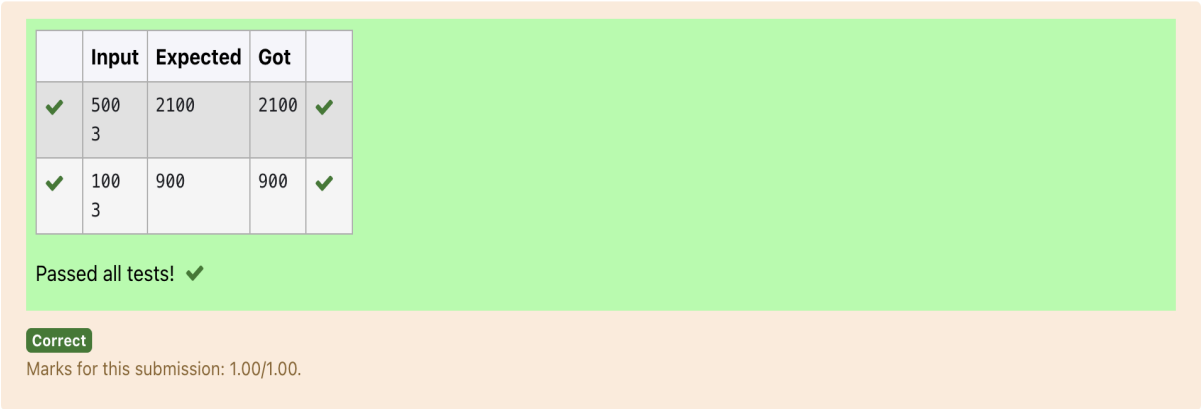
Output:
2100

Explanation:
On Monday the employee receives Rs.500, on Tuesday Rs.700, on WednesdayRs.900
So total = Rs.2100

CODE:

```
#include<stdio.h>
int main()
{
   int i,I,N,total;
   scanf("%d %d",&I,&N);
   for(i=0;i<N;i++)
   {
      total+=I;
      I=I+200;
   }
   printf("%d",total);
}
```

OUTPUT:



	Input	Expected	Got	
✔	500 3	2100	2100	✔
✔	100 3	900	900	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

6) Two numbers M and N are passed as the input. A number X is also passed as the input. The program must print the numbers divisible by X from N to M (inclusive of M and N).

Input Format:
The first line denotes the value of M
The second line denotes the value of N
The third line denotes the value of X

Output Format:
Numbers divisible by X from N to M, with each number separated by a space.
Boundary Conditions:
1 <= M <= 9999999
M < N <= 9999999
1 <= X <= 9999

Example Input/Output 1:
Input:
2
40
7

Output:
35 28 21 14 7

Example Input/Output 2:
Input:
66
121
11

Output:
121 110 99 88 77 66

CODE:

```
#include<stdio.h>
int main()
{
   int m,n,x,i;
   scanf("%d %d %d",&m,&n,&x);
   for(i=n;i>=m;i--)
   {
      if(i%x==0)
      {
         printf("%d ",i);
      }
   }
}
```

OUTPUT:

	Input	Expected	Got	
✔	2 40 7	35 28 21 14 7	35 28 21 14 7	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

7) Write a C program to find the quotient and reminder of given integers.

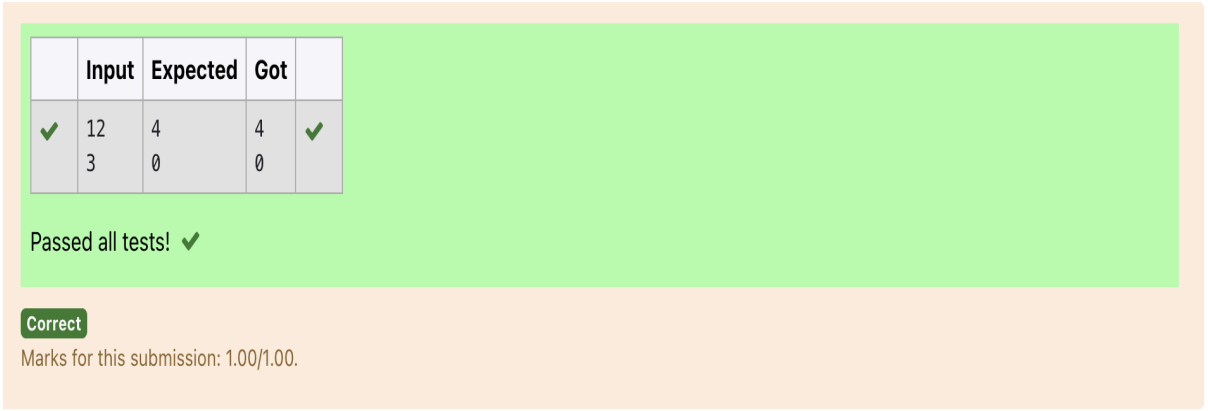
For example:

Input	Result
12 3	4 0

CODE:

```
#include<stdio.h>
int main()
{
   int n,d,q,r;
   scanf("%d %d",&n,&d);
   r=n%d;
   q=n/d;
   printf("%d\n",q);
   printf("%d",r);
}
```

OUTPUT:



	Input	Expected	Got	
✔	12 3	4 0	4 0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

8) Write a C program to find the biggest among the given 3 integers?

For example:

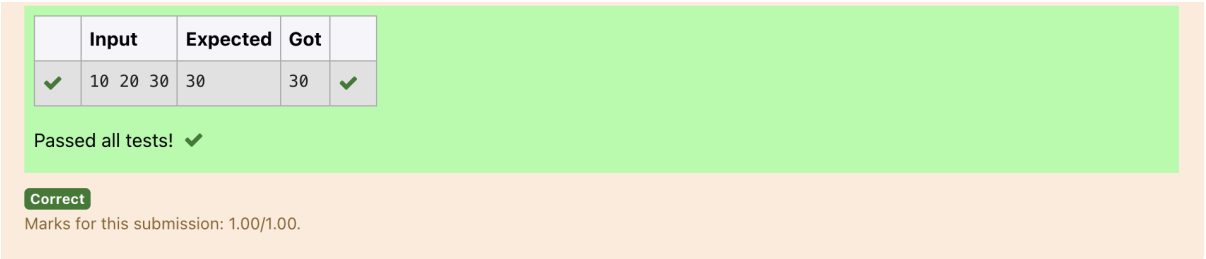
Input	Result
10 20 30	30

CODE:

```
#include<stdio.h>
int main()
{
```

```
    int a,b,c,result;
    scanf("%d %d %d",&a,&b,&c);
    if(a>b && a>c)
    {
       result=a;
    }
    else if(b>c)
    {
       result=b;
    }
    else
    {
       result=c;
    }
    printf("%d",result);
}
```

OUTPUT:



9) Write a C program to find whether the given integer is odd or even?

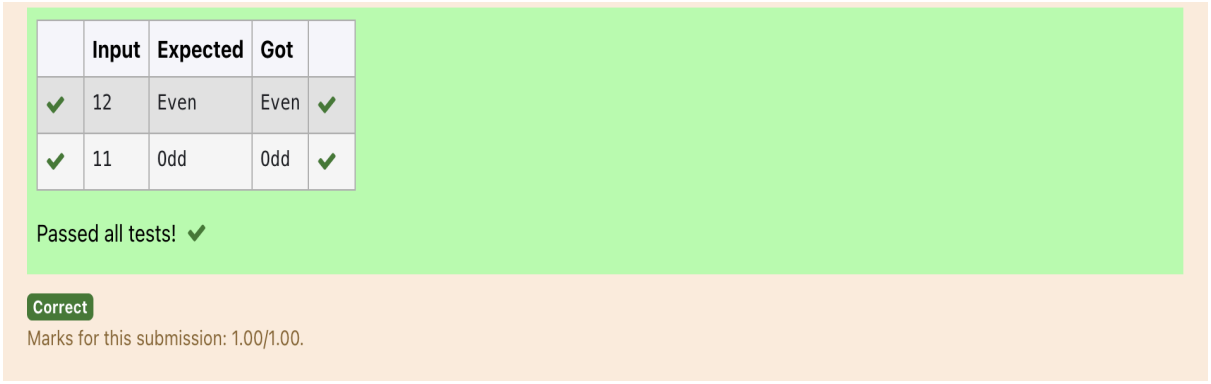
For example:

Input	Result
12	Even
11	Odd

CODE:

```
#include<stdio.h>
int main()
{
   int n;
   scanf("%d",&n);
   if(n%2==0)
      printf("Even");
   else
      printf("Odd");
}
```

OUTPUT:



	Input	Expected	Got	
✔	12	Even	Even	✔
✔	11	Odd	Odd	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

10) Write a C program to find the factorial of given n.

For example:

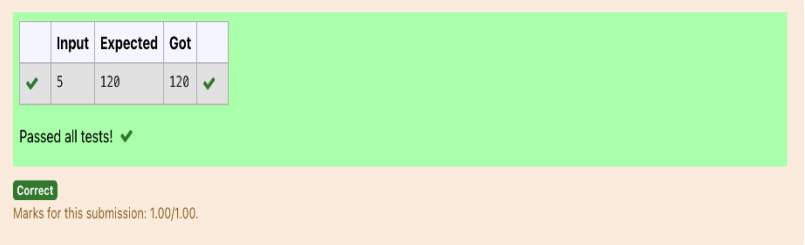
Input	Result
5	120

CODE:

```
#include<stdio.h>
int main()
{
```

```
    int a,i,fact=1;
    scanf("%d",&a);
    for(i=a;i>=1;i--)
        fact*=i;
    printf("%d",fact);
}
```

OUTPUT:



natural numbers.

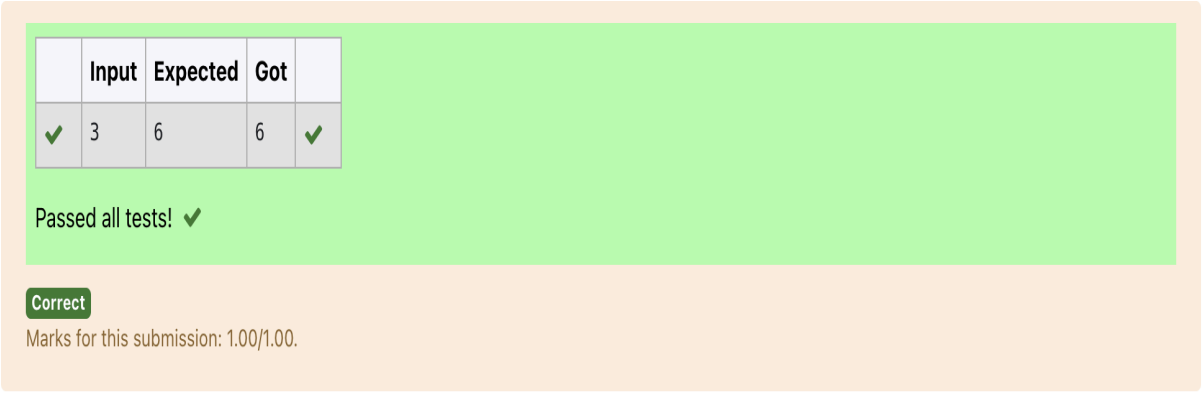
For example:

Input	Result
3	6

CODE:

```
#include<stdio.h>
int main()
{
    int n,i,sum;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        sum+=i;
    printf("%d",sum);
}
```

OUTPUT:



	Input	Expected	Got	
✔	3	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

12) Write a C program to find the Nth term in the fibonacci series.
For example:

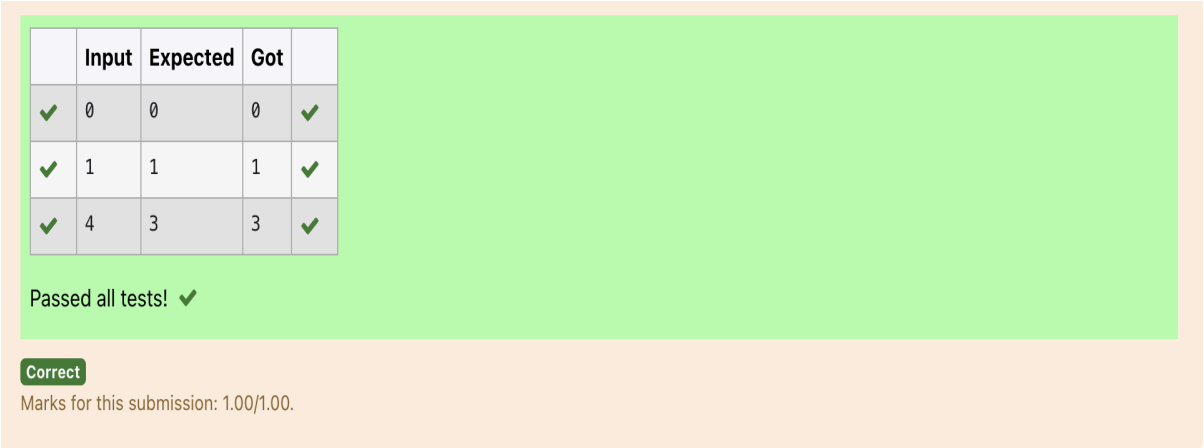
Input	Result
0	0
1	1
4	3

CODE:

```
#include<stdio.h>
int main()
{
   int n,c,a=1,b=1,i;
   scanf("%d",&n);
   if(n==0)
   {
      printf("0");
   }
   if (n==1 || n==2)
   {
      printf("1");
   }
   if(n>=3)
```

```
    {
        for(i=3 ; i<=n;i++)
        {
            c=a+b;
            a=b;
            b=c;
        }
        printf("%d",c);
    }
}
```

OUTPUT:



	Input	Expected	Got	
✔	0	0	0	✔
✔	1	1	1	✔
✔	4	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

13) Write a C program to find the power of integers.

input:
a b

output:
a^b value

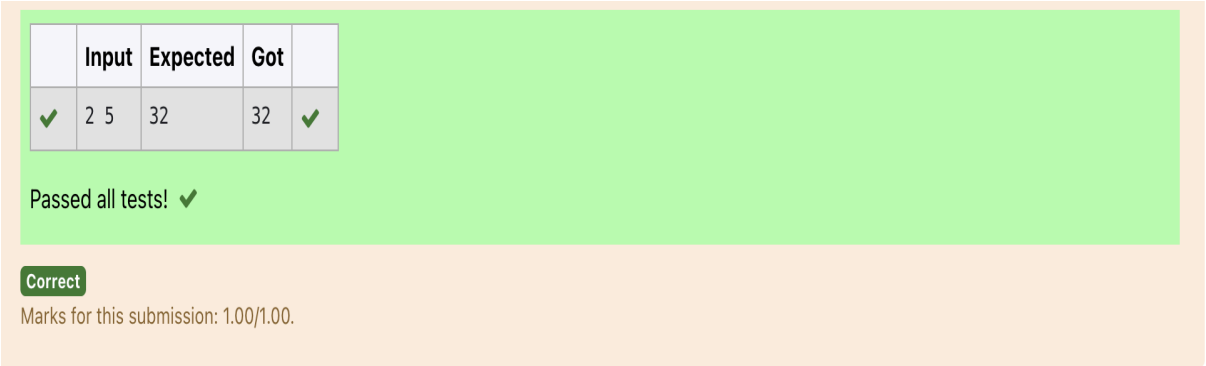
For example:

Input	Result
2 5	32

CODE:

```
#include<stdio.h>
#include<math.h>
int main()
{
   int a,p,r;
   scanf("%d %d",&a,&p);
   r=pow(a,p);
   printf("%d",r);
}
```

OUTPUT:



14) Write a C program to find Whether the given integer is prime or not.

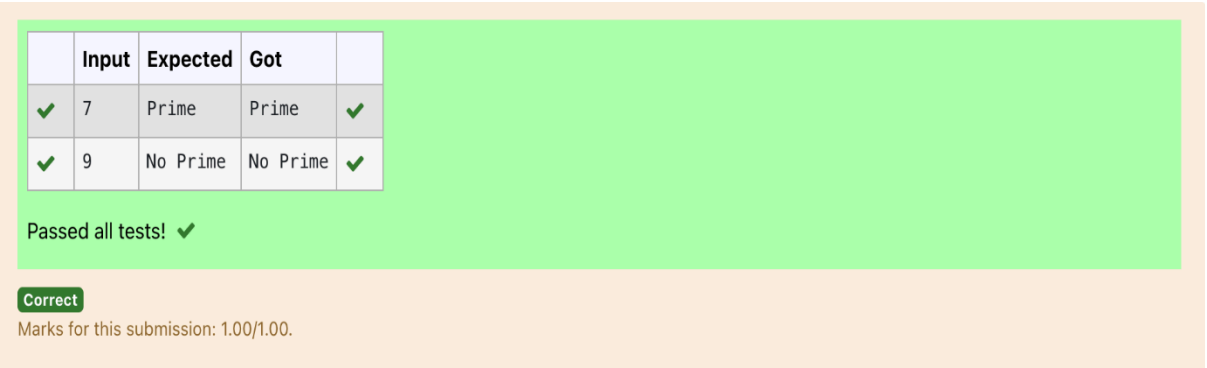
For example:

Input	Result
7	Prime
9	No Prime

CODE:

```
#include<stdio.h>
int main()
{
   int a,i,count=0;
   scanf("%d",&a);
   for(i=2;i<a;i++)
   {
      if(a%i==0)
         count++;
   }
   if(count==0)
      printf("Prime");
   else
      printf("No Prime");
}
```

OUTPUT:



	Input	Expected	Got	
✔	7	Prime	Prime	✔
✔	9	No Prime	No Prime	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

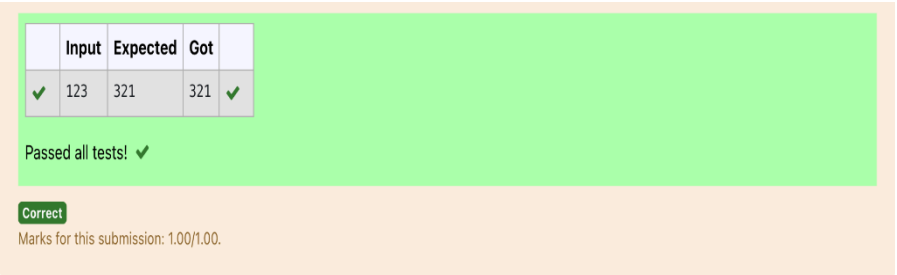
15) Write a C program to find the reverse of the given integer?

CODE:

```
#include<stdio.h>
int main()
{
```

```
    int sum=0,n,a,r;
    scanf("%d",&a);
    n=a;
    while(n!=0)
    {
        r=n%10;
        sum=(sum*10)+r;
        n=n/10;
    }
    printf("%d",sum);
}
```

OUTPUT:



WEEK 02

FINDING TIME COMPLEXITY OF ALGORITHMS

1) Convert the following algorithm into a program and find its time complexity using the counter method.

```
void function (int n)
{
   int i= 1;
   int s =1;
   while(s <= n)
   {
       i++;
       s += i;
    }
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
A positive Integer n
Output:
Print the value of the counter variable

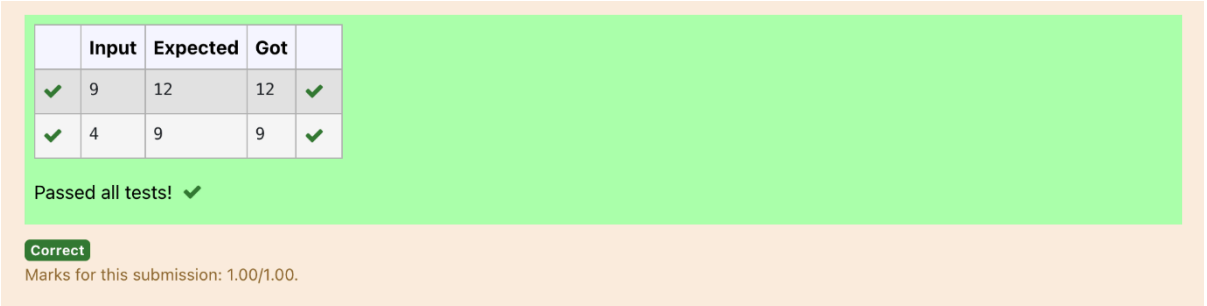
For example:

Input	Result
9	12

CODE:

```
#include<stdio.h>
void function (int n)
{
   int c=0;
   int i= 1;
   c++;
   int s =1;
   c++;
   while(s <= n)
   {
      c++;
       i++;
       c++;
       s += i;
       c++;
    }
    c++;
    printf("%d",c);
}
int main()
{
   int n;
   scanf("%d",&n);
   function(n);
}
```

OUTPUT:



	Input	Expected	Got	
✔	9	12	12	✔
✔	4	9	9	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2) Convert the following algorithm into a program and find its time complexity using the counter method.

```
void func(int n)
{
   if(n==1)
   {
     printf("*");
   }
   else
   {
    for(int i=1; i<=n; i++)
    {
      for(int j=1; j<=n; j++)
      {
         printf("*");
         printf("*");
         break;
      }
    }
  }
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

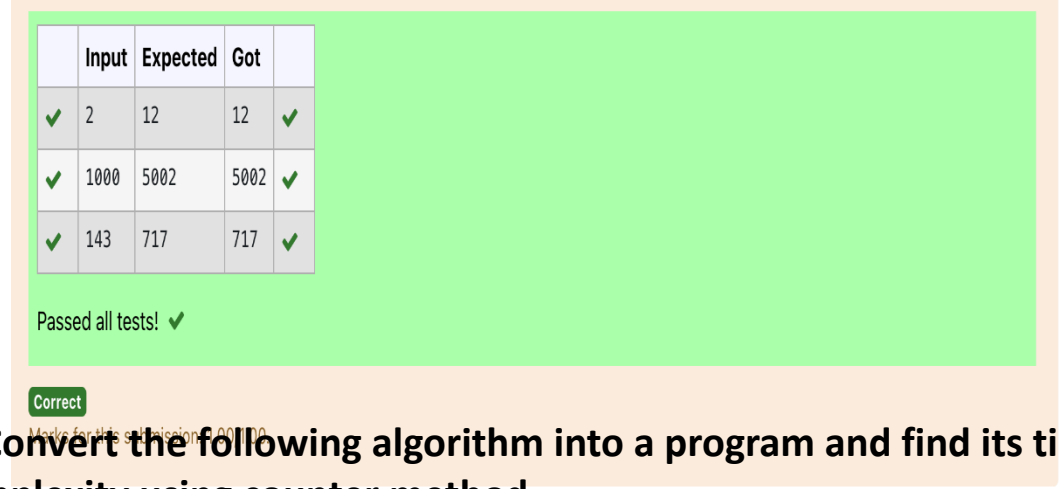
Print the value of the counter variable

CODE:

```
#include <stdio.h>
void func(int n)
{
   int c=0;
   if(n==1)
   {
     c++;
     printf("*");
     c++;
   }
   else
   {
    c++;
    for(int i=1; i<=n; i++)
    {
      c++;
      for(int j=1; j<=n; j++)
      {
        c++;
        //printf("*");
        c++;
        //printf("*");
        c++;
        break;
      }
      c++;
    }
    c++;
  }
  printf("%d",c);
}
int main()
{
```

```
    int n;
    scanf("%d",&n);
    func(n);
}
```

OUTPUT:



	Input	Expected	Got	
✔	2	12	12	✔
✔	1000	5002	5002	✔
✔	143	717	717	✔

Passed all tests! ✔

Correct

Marks for this submission: 20/20

3) Convert the following algorithm into a program and find its time complexity using counter method.

```
Factor(num) {
{
   for (i = 1; i <= num;++i)
   {
    if (num % i== 0)
      {
        printf("%d ", i);
      }
   }
 }
```

Note: No need of counter increment for declarations and scanf() and counter variable printf() statement.

Input:

A positive Integer n

Output:
Print the value of the counter variable

CODE:

```
#include<stdio.h>
void Factor(int num)
 {
   int c=0,i;

   for (i = 1; i <= num;++i)
   {
      c++;
      c++;
      if (num % i== 0)
      {

         //printf("%d ", i);
         c++;
      }

    }
    c++;
    printf("%d",c);

}
int main()
{
   int num;
   scanf("%d",&num);
   Factor(num);
  }
```

OUPUT:

	Input	Expected	Got	
✔	12	31	31	✔
✔	25	54	54	✔
✔	4	12	12	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4) Convert the following algorithm into a program and find its time complexity using counter method.

```
void function(int n)
{
   int c= 0;
   for(int i=n/2; i<n; i++)
      for(int j=1; j<n; j = 2 * j)
         for(int k=1; k<n; k = k * 2)
            c++;
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

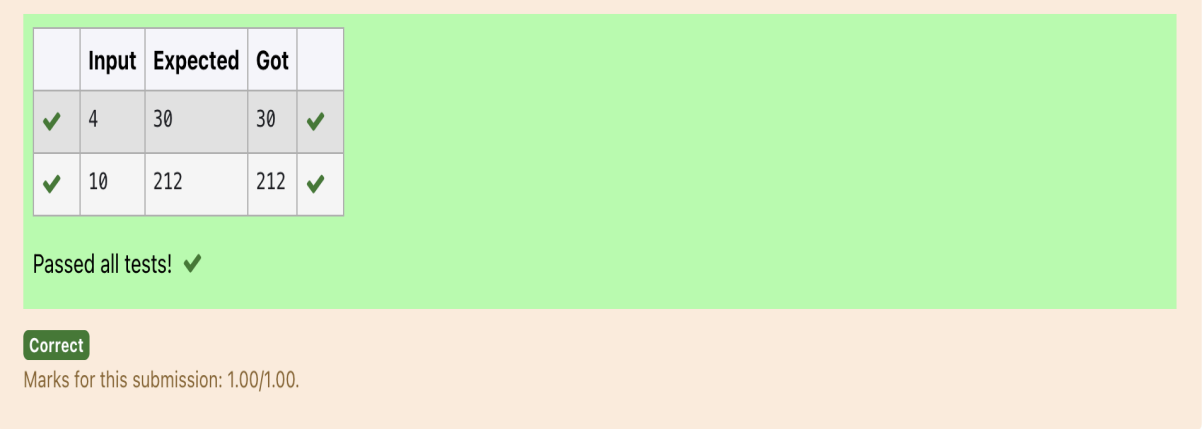
Input:
A positive Integer n
Output:
Print the value of the counter variable

CODE:

```
#include<stdio.h>
void function(int n)
{
   int count=0;
   int c= 0;
   count++;
   for(int i=n/2; i<n; i++)
   {
      count++;
      for(int j=1; j<n; j = 2 * j)
      {
         count++;
         for(int k=1; k<n; k = k * 2)
         {
            count++;
            c++;
            count++;
         }
         count++;
      }
      count++;
   }
   count++;
   printf("%d",count);
}

int main()
{
   int n;
   scanf("%d",&n);
   function(n);
}
```

OUTPUT:



	Input	Expected	Got	
✔	4	30	30	✔
✔	10	212	212	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5) Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
  int rev = 0, remainder;
  while (n != 0)
   {
      remainder = n % 10;
      rev = rev * 10 + remainder;
      n/= 10;

   }
print(rev);
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:
Print the value of the counter variable

CODE:

```
#include<stdio.h>
void reverse(int n)
{
   int count=0;
  int rev = 0, remainder;
  count++;
  while (n != 0)
   {
      count++;
      remainder = n % 10;
      count++;
      rev = rev * 10 + remainder;
      count++;
      n/= 10;
      count++;
   }
   count++;
   //printf("%d",rev);
   count++;
   printf("%d",count);
}

int main()
{
   int n;
   scanf("%d",&n);
   reverse(n);
}
```

OUTPUT:

	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

WEEK 03

DIVIDE AND CONQUER

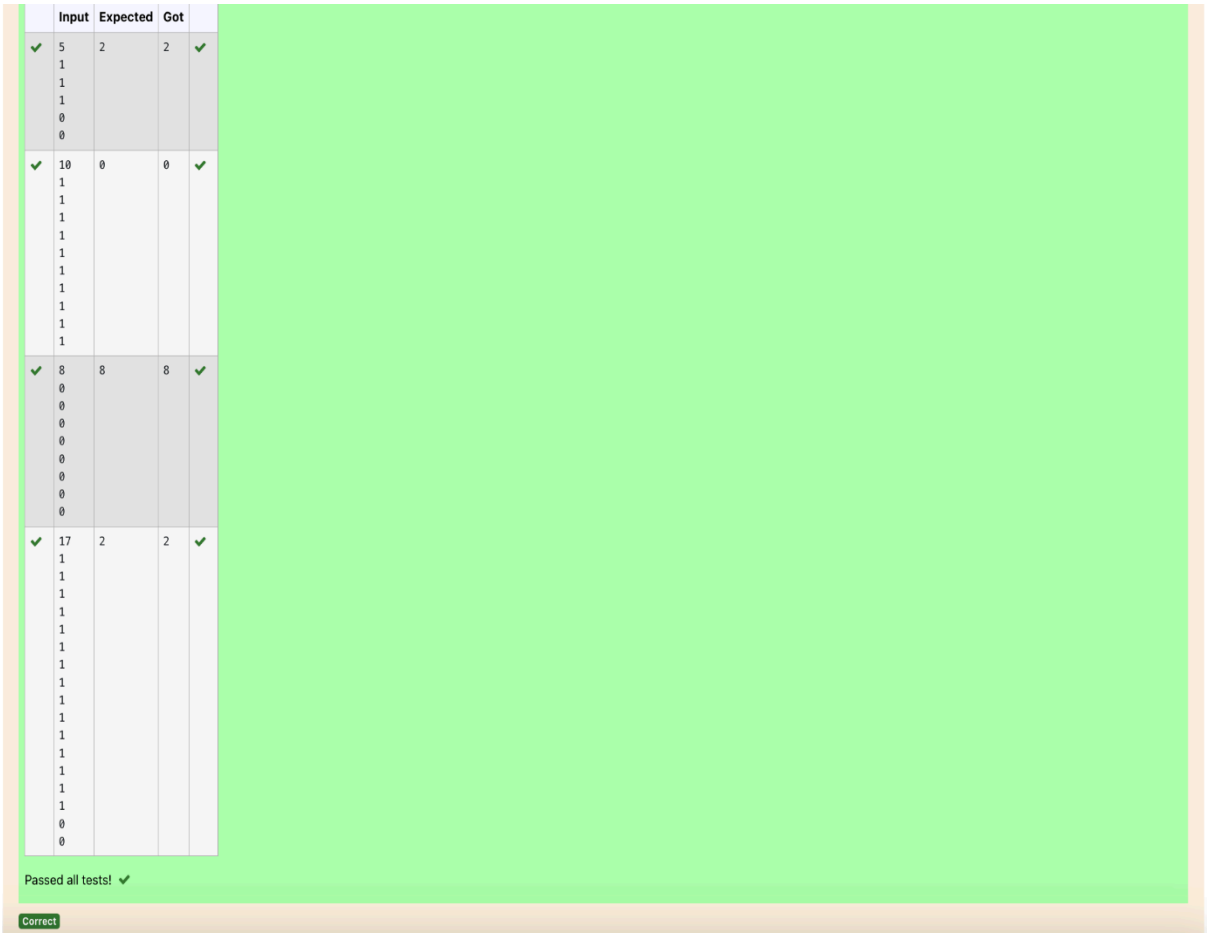
1) Problem Statement
Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.
Input Format:
First Line Contains Integer m – Size of array
Next m lines Contains m numbers – Elements of an array
Output Format:
First Line Contains Integer – Number of zeroes present in the given array.

CODE:

```
#include<stdio.h>
int conquer(int a[],int start,int end){
   int mid=(start+end)/2;
   if(start==end && a[start]==0){
      return 1;
   }
   if(start==end && a[start]!=0){
```

```
        return 0;
    }
    return(conquer(a,start,mid)+conquer(a,mid+1,end));
}
int main(){
    int n,i;
    scanf("%d",&n);
    int a[n];
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    int start=0,end=n-1;
    printf("%d",conquer(a,start,end));
}
```

OUTPUT:



2) Given an array nums of size n, return *the majority element*. The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Example 1:
Input: nums = [3,2,3]
Output: 3
Example 2:
Input: nums = [2,2,1,1,1,2,2]
Output: 2
Constraints:

- **n == nums.length**
- **$1 <= n <= 5 * 10^4$**
- **$-2^{31} <= nums[i] <= 2^{31} – 1$**

For example:

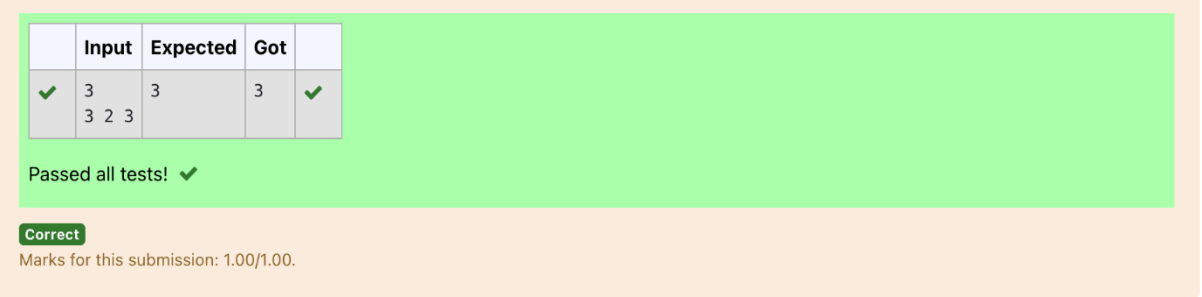
Input	Result
3 3 2 3	3
7 2 2 1 1 1 2 2	2

CODE:

```
#include<stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int a[n];
   for(int i=0;i<n;i++){
      scanf("%d",&a[i]);
   }
   for(int i=0;i<n;i++){
      int count=0;
      for(int j=0;j<n;j++){
```

```
            if(a[i]==a[j]){
                count++;
            }
        }
        if(count>n/2){
            printf("%d",a[i]);
            break;
        }
    }
}
```

OUTPUT:



conquer algorithm to find floor of x.

Input Format:

First Line Contains Integer n – Size of array
Next n lines Contains n numbers – Elements of an array
Last Line Contains Integer x – Value for x

Output Format:

First Line Contains Integer – Floor value for x

CODE:

```
#include<stdio.h>
int main(){
    int n,x,flr,i;
    scanf("%d",&n);
    int a[n];
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
```

```
    scanf("%d",&x);
    int mid=n/2;
    if(x<a[mid])
    {
        flr=a[0];
        for(i=0;i<mid;i++)
        {
            if(a[i]>=flr)
                if(a[i]<x)
                    flr=a[i];
        }
    }
    else
    {
        flr=a[mid];
        for(i=mid;i<n;i++)
        {
            if(a[i]>=flr)
                if(a[i]<x)
                    flr=a[i];
        }
    }
    printf("%d",flr);
}
```

OUTPUT:

	Input	Expected	Got	
✔	6 1 2 8 10 12 19 5	2	2	✔
✔	5 10 22 85 108 129 100	85	85	✔
✔	7 3 5 7 9 11	9	9	✔

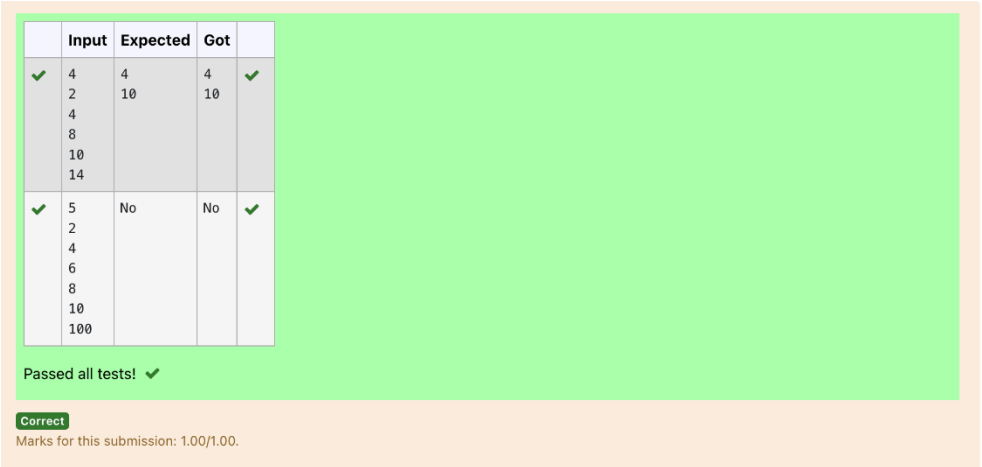
4) Problem Statement:
Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".
Note: Write a Divide and Conquer Solution
Input Format:
First Line Contains Integer n – Size of array
Next n lines Contains n numbers – Elements of an array
Last Line Contains Integer x – Sum Value
Output Format:
First Line Contains Integer – Element1
Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x").

CODE:

```
#include<stdio.h>
int main()
{
   int n,i,j,m,p,q,x;
   scanf("%d",&n);
   int a[n];
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
```

```
    scanf("%d",&x);
    for(i=0;i<n;i++)
    {
       for(j=i+1;j<n;j++){
          if((a[i]+a[j])==x){
             q=a[i]+a[j];
             m=a[i];
             p=a[j];
          }
       }
    }
    if(q==x) {
       printf("%d\n",m);
       printf("%d",p);
    }
    else
       printf("No");
}
```

OUTPUT:



	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔
✔	5 2 4 6 8 10 100	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5) Write a Program to Implement the Quick Sort Algorithm

Input Format:
The first line contains the no of elements in the list-n
The next n lines contain the elements.

Output:
Sorted list of elements

For example:

Input	Result
5 67 34 12 98 78	12 34 67 78 98

CODE:

```
#include<stdio.h>
int main()
{
   int n,i,j,temp;
   scanf("%d",&n);
   int a[n];
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   int x;
   scanf("%d",&x);
   for(i=0;i<n;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(a[i]>a[j])
         {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
         }
      }
   }
    for(i=0;i<n;i++)
```

```
        printf("%d ",a[i]);
}
```

OUTPUT:

	Input	Expected	Got	
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✔
✔	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	✔
✔	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

WEEK 04
GREEDY ALGORITHMS

1) Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.
Input Format:
Take an integer from stdin.
Output Format:
print the integer which is change of the number.
Example Input :
64
Output:
4
Explanaton:
We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

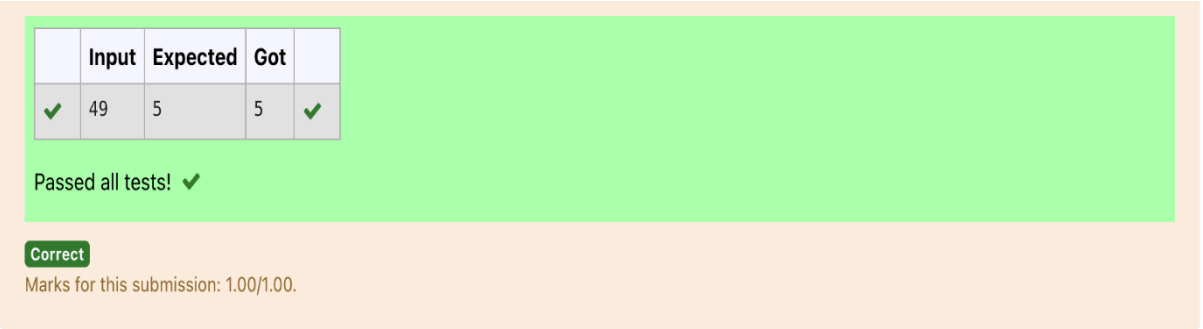
CODE:

```
#include <stdio.h>
int main()
{
   int cost;
```

```
    scanf("%d",&cost);
    int coin[9] = {1,2,5,10,20,50,100,500,1000};
    int i=0, count= 0;

    for (i=9-1; i>0; i--) {
        while (cost >= coin[i]) {
            cost -= coin[i];
            count++;
        }
    }
    printf("%d",count);
}
```

OUTPUT:



2) Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.
Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.
Example 1:
Input:
3
1 2 3

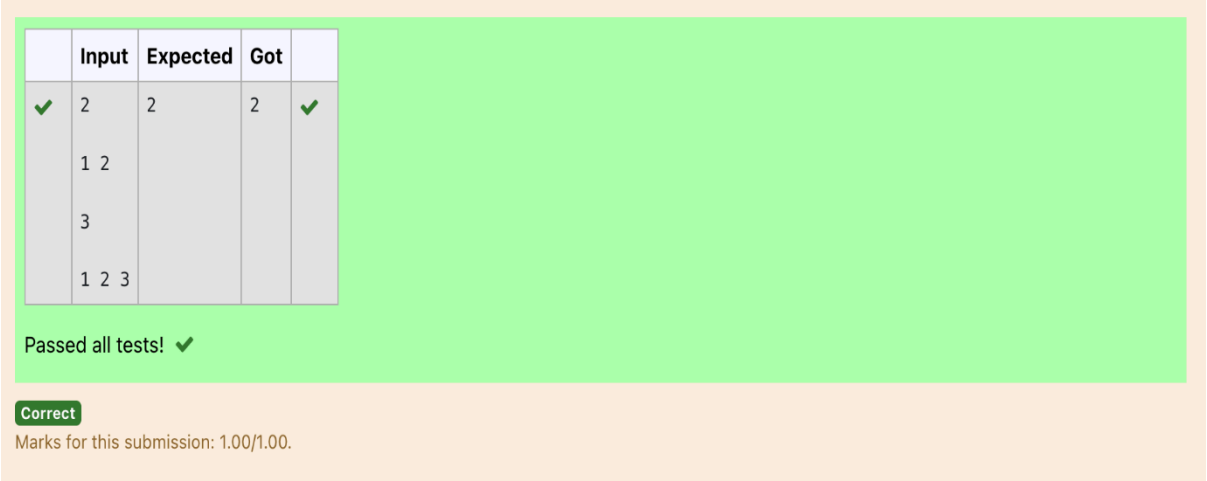
2
1 1
Output:
1
Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.
And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.
You need to output 1.
Constraints:
1 <= g.length <= 3 * 10^4
0 <= s.length <= 3 * 10^4
1 <= g[i], s[j] <= 2^31 – 1

CODE:

```
#include<stdio.h>
int main(){
   int chno,cono;
   int satisfied=0,j=0;
   scanf("%d",&chno);
   int child[chno];
   for(int i = 0;i<chno;i++){
      scanf("%d",&child[i]);
   }
   scanf("%d",&cono);
   int cookie[cono];
   for(int i = 0; i<cono;i++){
      scanf("%d",&cookie[i]);
   }
   for(int i=0;i<chno;i++){
      if(child[i]<=cookie[j]){
         satisfied+=1;
         j++;
      }
   }
```

```
    printf("%d",satisfied);
}
```

OUTPUT:



	Input	Expected	Got	
✔	2 1 2 3 1 2 3	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3) A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories.

If he has eaten *i* burgers with c calories each, then he has to run at least $3^i * c$ kilometers to burn out the calories. For example, if he ate 3

burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$.

But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance

he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is n space-separate integers

Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

3
5 10 7

Sample Output
76

For example:

Test	Input	Result
Test Case 1	**3** **1 3 2**	**18**

CODE:

```
#include<stdio.h>
#include<math.h>
int main()
{
   int burg,i,j;
   scanf("%d",&burg);
   int cal[burg];
   for(i=0;i<burg;i++){
      scanf("%d",&cal[i]);
   }
   int temp,kms=0;
   for(i=0;i<burg-1;i++){
      for(j=0;j<burg-i-1;j++){
         if(cal[j]>cal[j+1]){
            temp=cal[j];
            cal[j]=cal[j+1];
```

```
            cal[j+1]=temp;
          }
        }
    }
    j=burg;
    for(i=0;i<burg;i++){
        kms+=(pow(burg,i)*cal[j-1]);
        j--;
    }
    printf("%d",kms);
    return 0;
}
```

OUTPUT:

	Test	Input	Expected	Got	
✔	Test Case 1	3 1 3 2	18	18	✔
✔	Test Case 2	4 7 4 9 6	389	389	✔
✔	Test Case 3	3 5 10 7	76	76	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4) Given an array of N integer, we have to maximize the sum of arr[i] * i, where i is the index of the element (i = 0, 1, 2, ..., N).Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

2 5 3 4 0

Sample output:

40

CODE:

```
#include<stdio.h>
int main()
{
   int N,temp,i,sum=0;
   scanf("%d",&N);
   int arr[N];
   for(i=0;i<N;i++){
      scanf("%d",&arr[i]);
   }
   for (int i = 0; i < N - 1; i++) {
      int mind = i;
      for (int j = i + 1; j < N; j++) {
         if (arr[j] < arr[mind]) {
            mind = j;
         }
      }
      temp = arr[mind];
      arr[mind] = arr[i];
      arr[i] = temp;
   }
   for(i=0;i<N;i++){
      sum=sum+arr[i]*i;
   }
   printf("%d",sum);
   return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✔	5 2 5 3 4 0	40	40	✔
✔	10 2 2 2 4	191	191	✔

5) Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

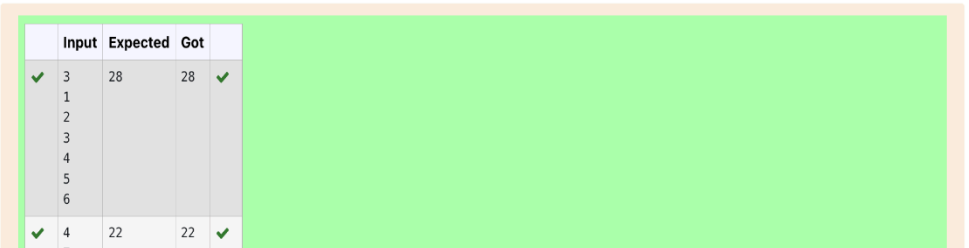
Input	Result
3 1 2 3 4 5 6	28

CODE:

```
#include<stdio.h>
int main()
{
   int N,i,j,temp,sum,flag=0;
   scanf("%d",&N);
```

```
    int arr1[N];
    int arr2[N];
    for(i=0;i<N;i++){
        scanf("%d",&arr1[i]);
    }
    for(i=0;i<N;i++){
        scanf("%d",&arr2[i]);
    }
    for (i=0;i<N-1;i++){
        for(j =0;j<N-i-1;j++){
            if(arr1[j]>arr1[j+1]){
                temp=arr1[j];
                arr1[j]=arr1[j+1];
                arr1[j+1]=temp;
            }
            if(arr2[j]>arr2[j+1]){
                temp=arr2[j];
                arr2[j]=arr2[j+1];
                arr2[j + 1]=temp;
            }
        }
    }
    i=0;
    j=N-1;
    while(flag<N){
        sum+=arr1[i]*arr2[j];
        i++;
        j--;
        flag++;
    }
    printf("%d",sum);
}
```

OUTPUT:



	Input	Expected	Got	
✔	3 1 2 3 4 5 6	28	28	✔
✔	4	22	22	✔

WEEK 05
DYNAMIC PROGRAMMING

1) Playing with Numbers:
Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.
Example 1:

Input: 6
Output:6
Explanation: There are 6 ways to 6 represent number with 1 and 3
1+1+1+1+1+1
3+3
1+1+1+3
1+1+3+1
1+3+1+1
3+1+1+1
Input Format
First Line contains the number n

Output Format

Print: The number of possible ways ‘n’ can be represented using 1 and 3

Sample Input

6

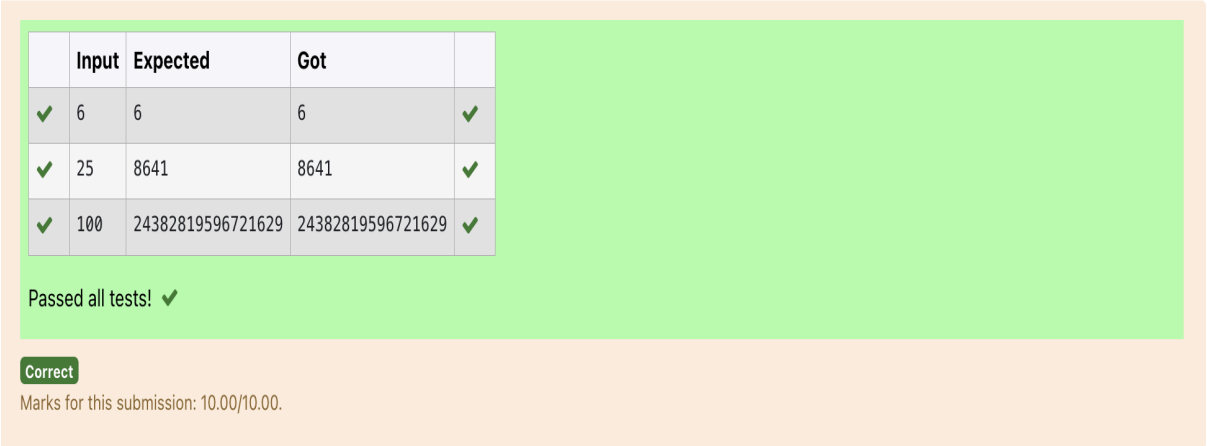
Sample Output

6

CODE:

```
#include <stdio.h>
#include <stdlib.h>
long long countWays(int n) {
    if (n < 0) return 0;
    if (n == 0) return 1;
    long long *dp = (long long*)calloc(n + 1, sizeof(long long));
    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 1;
    for (int i = 3; i <= n; i++) {
        dp[i] = dp[i-1] + dp[i-3];
    }
    long long result = dp[n];
    free(dp);
    return result;
}
int main() {
    int n;
    scanf("%d", &n);
    long long ways = countWays(n);
    printf("%lld\n", ways);
    return 0;
}
```

OUTPUT:



	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	24382819596721629	24382819596721629	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

2) Playing with Chessboard:

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

Example:

Input

3

1 2 4

2 3 4

8 7 1

Output:

19

Explanation:

Totally there will be 6 paths among that the optimal is

Optimal path value:1+2+8+7+1=19

Input Format

First Line contains the integer n

The next n lines contain the n*n chessboard values

Output Format
Print Maximum monetary value of the path

CODE:

```
#include <stdio.h>
#include <stdlib.h>
int max(int a, int b) {
   return (a > b) ? a : b;
}
int findMaxPath(int n, int **board) {
   int **dp = (int **)malloc(n * sizeof(int *));
   for (int i = 0; i < n; i++) {
      dp[i] = (int *)malloc(n * sizeof(int));
   }
   dp[0][0] = board[0][0];
   for (int j = 1; j < n; j++) {
      dp[0][j] = dp[0][j-1] + board[0][j];
   }
   for (int i = 1; i < n; i++) {
      dp[i][0] = dp[i-1][0] + board[i][0];
   }
   for (int i = 1; i < n; i++) {
      for (int j = 1; j < n; j++) {
         dp[i][j] = max(dp[i-1][j], dp[i][j-1]) + board[i][j];
      }
   }
   int result = dp[n-1][n-1];
   for (int i = 0; i < n; i++) {
      free(dp[i]);
   }
   free(dp);
   return result;
}
```

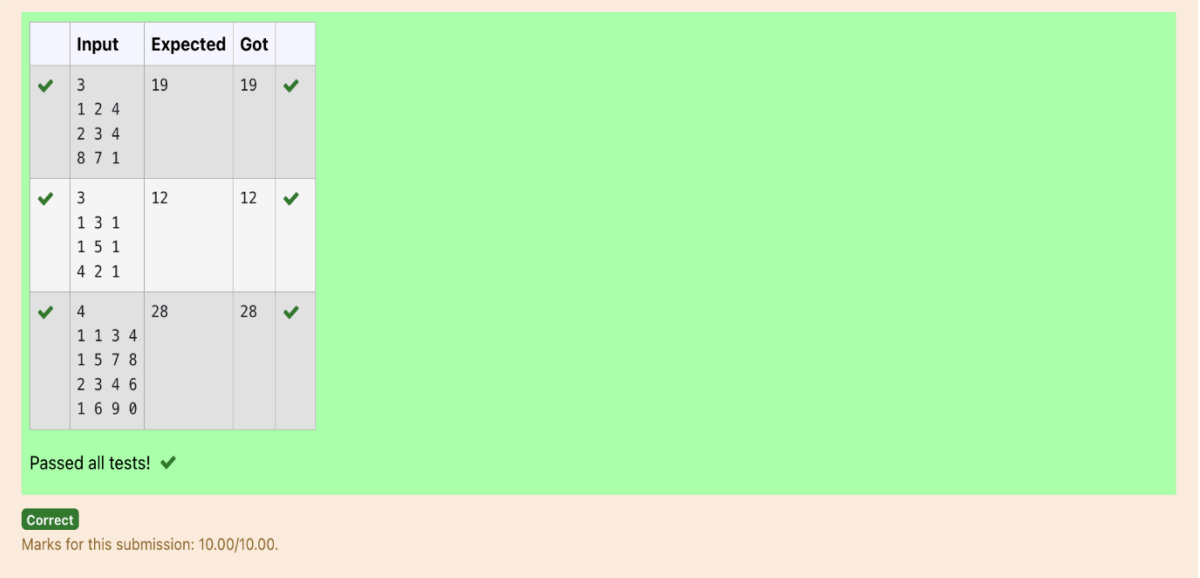
```
int main() {
  int n;
   scanf("%d", &n);

   int **board = (int **)malloc(n * sizeof(int *));
   for (int i = 0; i < n; i++) {
      board[i] = (int *)malloc(n * sizeof(int));
      for (int j = 0; j < n; j++) {
         scanf("%d", &board[i][j]);
      }
   }

   int maxPath = findMaxPath(n, board);
   printf("%d\n", maxPath);

   // Free the board array
   for (int i = 0; i < n; i++) {
      free(board[i]);
   }
   free(board);
   return 0;
}
```

OUTPUT:



	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔
✔	3 1 3 1 1 5 1 4 2 1	12	12	✔
✔	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

3) Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:
s1: ggtabe
s2: tgatasb

s1		**a**	**g**	**g**	**t**	**a**	**b**
s2	**g**	**x**	**t**	**x**	**a**	**y**	**b**

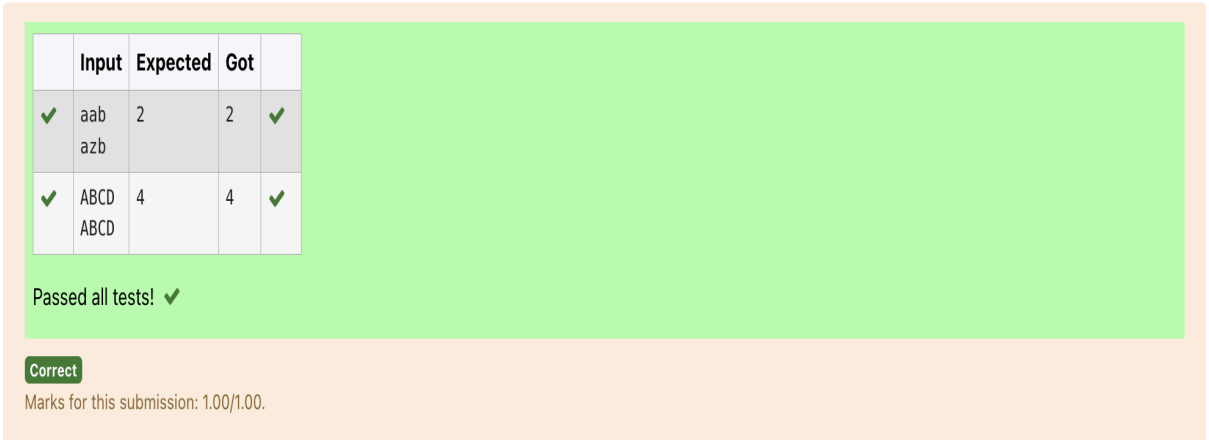
The length is 4
Solving it using Dynamic Programming
For example:

Input	Result
aab azb	2

CODE:

```
#include <stdio.h>
#include <string.h>
int lcs(char *s1, char *s2) {
   int m = strlen(s1);
   int n = strlen(s2);
   int dp[m+1][n+1];
   for (int i = 0; i <= m; i++) {
      for (int j = 0; j <= n; j++) {
         if (i == 0 || j == 0) {
            dp[i][j] = 0;  // LCS of any string with an empty string is 0
         }
         else if (s1[i-1] == s2[j-1]) {
            dp[i][j] = dp[i-1][j-1] + 1;  // Characters match
         }
         else {
            dp[i][j] = (dp[i-1][j] > dp[i][j-1]) ? dp[i-1][j] : dp[i][j-1
         }
      }
   }
     return dp[m][n];
}
int main() {
   char s1[100], s2[100];
   scanf("%s %s", s1, s2);
   int result = lcs(s1, s2);
   printf("%d\n", result);
   return 0;
}
```

OUTPUT:



	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4) Problem statement:
Find the length of the Longest Non-decreasing Subsequence in a given Sequence.
Eg:

Input:9
Sequence:[-1,3,4,5,2,2,2,2,3]
the subsequence is [-1,2,2,2,2,3]
Output:6

CODE:

```
#include<stdio.h>
int main()
{
   int n,i,j;
   scanf("%d",&n);
   int a[n];
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   int b[n];
   for(i=0;i<n;i++)
     b[i]=1;
   int max=1;
   for (i = 1; i < n; i++) {
      for (j = 0; j < i; j++) {
```

```
            if (a[j] <= a[i]) {
                b[i] = b[i] > (b[j] + 1) ? b[i] : (b[j] + 1);
            }
        }
        if(b[i]>max)
            max=b[i];
    }
    printf("%d",max);
}
```

OUTPUT:

	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

WEEK 06
COMPETITIVE PROGRAMMING

1) Find Duplicate in Array.
Given a read only array of n integers between 1 and n, find one number that repeats.
Input Format:

First Line - Number of elements
n Lines - n Elements
Output Format:
Element x - That is repeated
For example:

Input	Result
5 1 1 2 3 4	1

CODE:

```
#include<stdio.h>
int main()
{
   int n,i,j;
   scanf("%d",&n);
   int a[n];
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   for(i=0;i<n;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(a[i]==a[j])
            printf("%d",a[i]);
      }
   }
}
```

OUTPUT:

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2) Find the intersection of two sorted arrays.
OR in other words,
Given 2 sorted arrays, find all the elements which occur in both the arrays.
Input Format
· The first line contains T, the number of test cases. Following T lines contain:
1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array
Output Format
The intersection of the arrays in a single line
Example
Input:
1
3 10 17 57
6 2 7 10 15 57 246
Output:
10 57
Input:
1
6 1 2 3 4 5 6
2 1 6
Output:
1 6

For example:

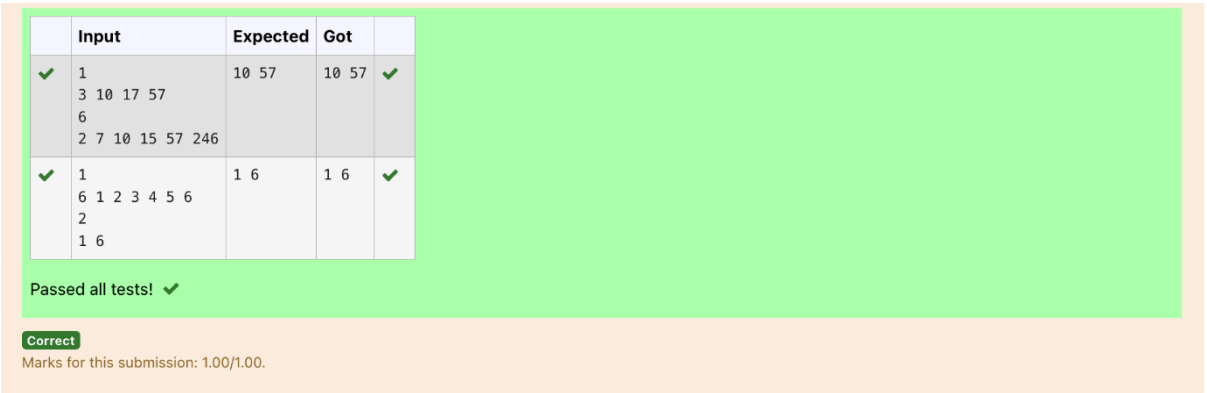
Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

CODE:

```
#include <stdio.h>
int main() {
    int t, n1, n2, i, j;
    scanf("%d", &t);
    while (t--) {
        scanf("%d", &n1);
        int a[n1];
        for (i = 0; i < n1; i++)
            scanf("%d", &a[i]);
        scanf("%d", &n2);
        int b[n2];
        for (j = 0; j < n2; j++) {
            scanf("%d", &b[j]);
        }
        i=0;
        j=0;
        while(i<n1 &&j<n2)
        {
            if(a[i]==b[j])
            {
                printf("%d ",a[i]);
                i++;
                j++;
            }
            else if(a[i]<b[j])
                i++;
```

```
            else
                j++;
        } }
}
```

OUTPUT:



	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3) Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

CODE:

```
#include <stdio.h>
int main() {
   int n, k, i, j;
   scanf("%d", &n);
   int a[n];
   for(i = 0; i < n; i++)
   {
      scanf("%d", &a[i]);
   }
   scanf("%d", &k);
   for(i = 0; i < n; i++) {
      for(j = i + 1; j < n; j++)
      {
         if(a[j] - a[i] == k)
         {
            printf("1\n");
            return 0;
         }
      }
   }
   printf("0\n");
}
```

OUTPUT:

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.