NAME:TEJASWIN.G

ROLL NO:230701359

CSE-F

DESIGN OF ANALYSIS AND ALGORITHMS

WEEK 1: BASIC C - PROGRAMMING PRACTICE

PROGRAM 1:

AIM: Given 2 numbers, write a program to swap them.

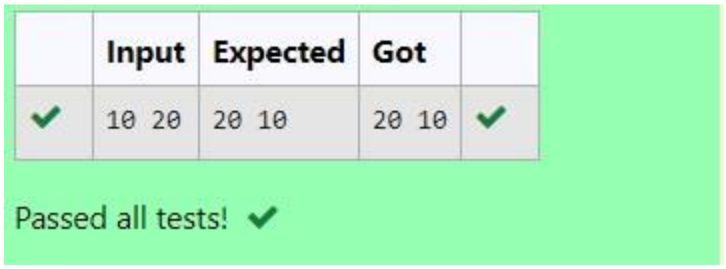
ALGORITHM:

```
Step 1: Initialize a,b,temp as int
Step 2: Input numbers from user for a and
b Step 3: Perform temp=a, a=b, b=temp
Step 4: Display the number
```

PROGRAM:

```
#include<stdio.

h> int main()
 {
    int a,b,temp; scanf("%d
    %d",&a,&b); temp=a;
    a=b;
    b=tem
    p; printf("%d
    %d",a,b);
 }
```

OUTPUT:



	Input	Expected	Got	
✓	`10 20`	`20 10`	`20 10`	✓

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

PROGRAM

2:

AIM: Write a program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Math >= 65

Marks in Physics >= 55 [or] Total in all subjects >=180 Marks in Chemistry >= 50

ALGORITHM:

Step 1: Initialize m as math, p as physics, c as chemistry all as int datatype.
Step 2: Input 3 numbers out of 100 from the user.

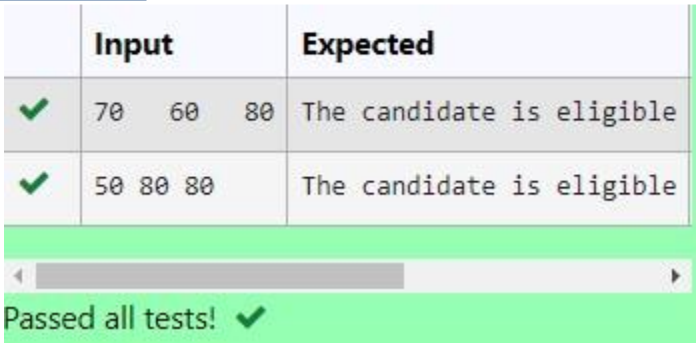
Step 3: Check if m>=65 and p>=55 and c>=50 →Then display "the candidate is eligible" Or check if m+p+c>=180 → Then display "the candidate is eligible"

Else → Display "the candidate is not eligible"

PROGRAM:

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

 {

    int m,p,c;
    scanf("%d%d%d",&m,&
    p,&c);

    if (m>=65 && p>=55 && c>=50){
       printf("The candidate is eligible");

    }else if(m+p+c>=180){ printf("The

       candidate is eligible");

    }else{

       printf("The candidate is not eligible");

    }}
```

OUTPUT:



	Input	Expected
✓	70 60 80	The candidate is eligible
✓	50 80 80	The candidate is eligible

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

PROGRAM

______________3:

AIM: Malini goes to Best save hyper market to buy grocery items. Bestsave hypermarket provides 10% discount on the bill amount B whenever the bill amount B is more than Rs. 2000. The bill amount B is passed as the input to the program and it must print the final amount payable by Malini.

ALGORITHM:

Step 1: Initialize the payment and the discount as integer data types.
Step 2: Take an input for payment from the user.

Step 3: Check if payment > 2000, → calculate discount as payment*0.10 and subtract it from the original payment amount.

Display the new payment.

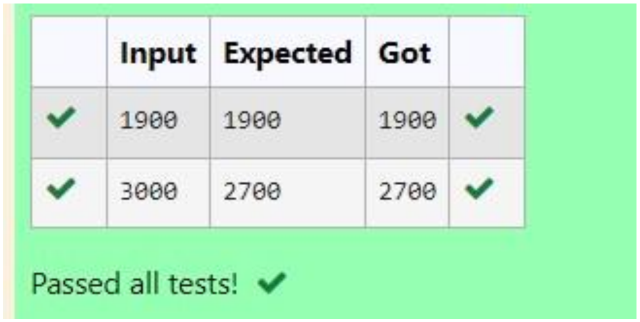
Step 4: Else → display the payment amount.

PROGRAM:

```
#include<stdi
o.h> int main()

{

   int pay,disc; scanf("%d",&p
   ay); if
   (pay>2000){

      disc=pay*0.1 0;
      pay=paydisc;
      printf("%d",p
      ay);

   }else{ printf("%d",pay);

   }
```

}

OUTPUT:



	Input	Expected	Got	
✓	1900	1900	1900	✓
✓	3000	2700	2700	✓

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

4:

AIM: **Baba is very kind to beggars and every day Baba donates half of the amount he has whenever 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 at the beginning of the day.**

ALGORITHM:

Step 1: Initialize m and n as integer data types symbolizing the money and the number of beggars.

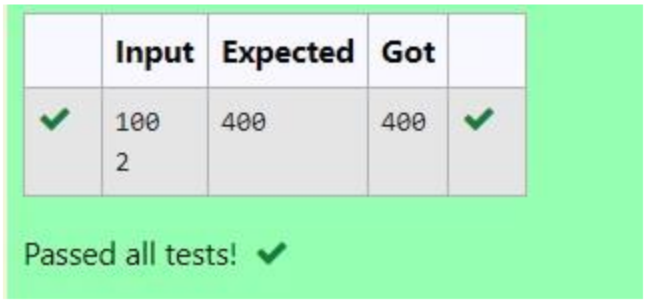
Step 2: Take an input from the user for the number of beggars and the money amount. Step 3: Initialize the for loop until n, and multiply the money as money=money * n Step 4: Outside the loop display the amount m symbolizing the money in hand.

PROGRAM:

PROGRAM

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

OUTPUT:



	Input	Expected	Got	
✓	100 2	400	400	✓

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

______________5:

AIM: **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 [starting from Monday through Saturday] he will be awarded Rs. 200 more than the previous day as "Punctuality incentive". Incentive for starting day is passed as input and the number of days N is also passed. The program is to calculate the "Punctuality incentive" P of the employee.**

ALGORITHM:

Step 1: Initialize incentive i, n number of days and sum as integer datatype
Step 2: Take an input from the user for incentive and number of days i and n. Step 3: initialize the sum as i, and initiate a for loop till n-1;

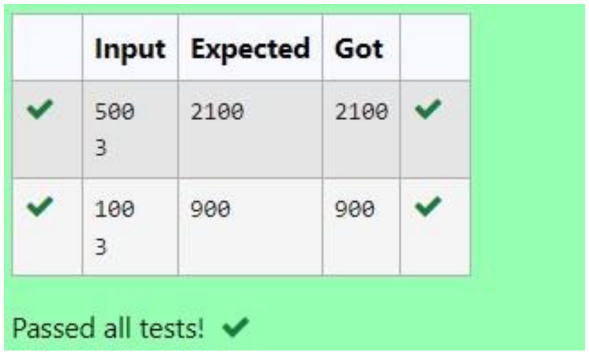
Within this for loop, calculate incentive as incentive + 200 and the sum + incentive. Step 4: Outside the loop, display the sum.

PROGRAM:

```
#include<stdi

o.h> int main()

{

   int              i,n,sum;
   scanf("%d%d",&i,
   &n); sum=i;

   for (int j=1;j<n;j++){
      i=i+200; sum+=i;
      }printf("%d",sum);

}
```

PROGRAM

OUTPUT:



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

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

6:

AIM: Two numbers a and b are passed as the input. A number x is also passed as the input. The program must print the numbers divisible by x from b to a range inclusive of a and b.

ALGORITHM:

Step 1: Initialize the numbers as a, b, c as integer data types.
Step 2: Take an input for a, b and c from the user.

Step 3: In a for loop, >=a, decrementing the value, Check if i%c==0, → Display the number i

Else → continue

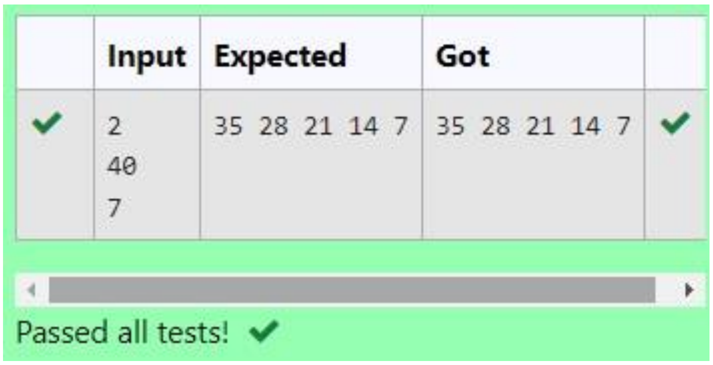
PROGRAM:

```
#include<stdi
o.h> int main()

{

   int a,b,c;
   scanf("%d%d%d",&a,&b
   ,&c); for (int i=b;i>=a;i--)

   {

      if(i%c==0)

      {

         printf("%d ",i);

      }

      else continu
      e;
```

PROGRAM

```
   }
}
```

OUTPUT:



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

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

_____________7:

AIM: Write a program to find the quotient and remainder of the given integers.

ALGORITHM:

Step 1: Initialize the 2 numbers a and b.

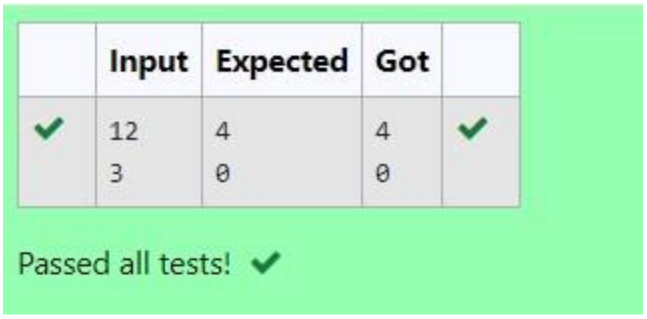
Step 2: Take an input for a and b from the user.
Step 3: Display a/b and a%b.

PROGRAM:

```
#include<stdi
o.h> int main()
{ int a,b;
```

```
    scanf("%d%d",&a,
    &b); printf("%d\n",a/b);

    printf("%d",a%b);

}
```

OUTPUT:



	Input	Expected	Got	
✓	12 3	4 0	4 0	✓

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

PROGRAM

______________8:

AIM: **Write a program to find the biggest number out of the 3 given integers.**

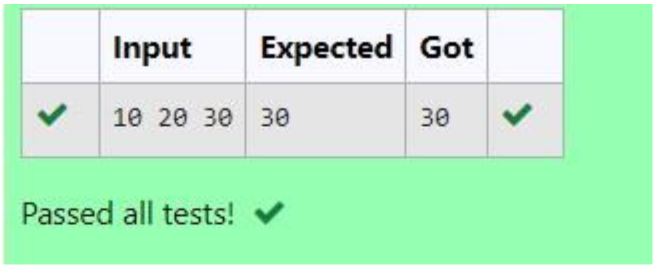
ALGORITHM:

```
Step 1: Initialize the 3 numbers as a, b, c as integer data types.
Step 2: Take an input from the a, b, c.
Step 3: Check if a>b and a>c → Display a
        Else check if b>a and b>c → Display
        b Else check if c>a and c>b →
        Display c
```

PROGRAM:

```
#include<stdi
o.h> int main()
{
   int a,b,c;
   scanf("%d%d%d",&a,&b
   ,&c); if (a>b && a>c)
      printf("%d",a) ;
   else if (b>a &&
   b>c)
      printf("%d",b)
   ; else if (c>a
   && c>b) printf("%d",c);
}
```

OUTPUT:



	Input	Expected	Got	
✓	10 20 30	30	30	✓

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

PROGRAM

program to find

9:

AIM: Write a C

ALGORITHM:

whether the given number is odd or even.

Step 1: Initialize a number M as integer data type.

Step 2: Take an input from the user.

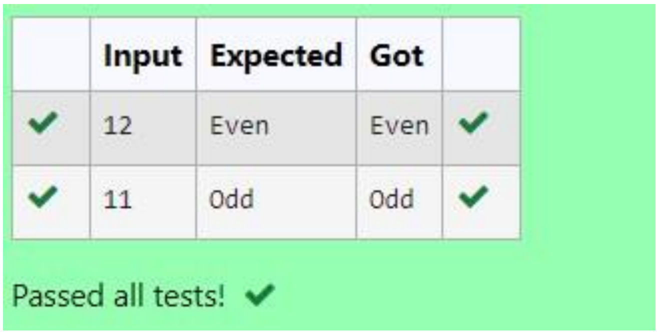
Step 3: Check if m%2==0 → Display even Else

→ Display odd.

PROGRAM:

```
#include<stdi
o.h> int main()

{

   int m; scanf("%d",&
   m); if
   (m%2==0)
   printf("Even" );
   else
   printf("Odd");

}
```

OUTPUT:



	Input	Expected	Got	
✓	12	Even	Even	✓
✓	11	Odd	Odd	✓

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

PROGRAM

program to find the

10:

AIM: Write a C

ALGORITHM:

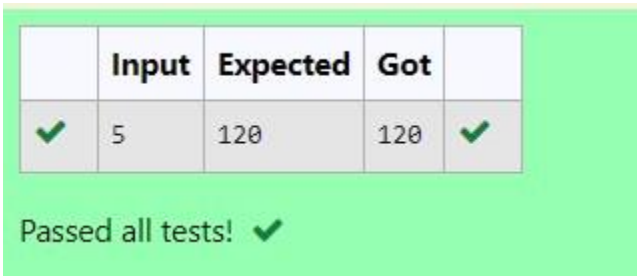
factorial of a number N.

```
Step 1: Initialize x , i and factorial=1 as integer data type.
Step 2: Take an input for x.
Step 3: In a for loop, as i=1, and i<=x
        Calculate fact*=i
Step 4: Display the factorial.
```

PROGRAM:

```
#include<stdi
o.h> int main()
{
   int x,i,fact=1; scanf("%d",&
   x); for
   (i=1;i<=x;i++)
      fact*=i;
   printf("%d",f act);
}
```

OUTPUT:



	Input	Expected	Got	
✓	5	120	120	✓

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

PROGRAM

program to find the

11:

AIM: Write a C

ALGORITHM:

sum of first N natural.

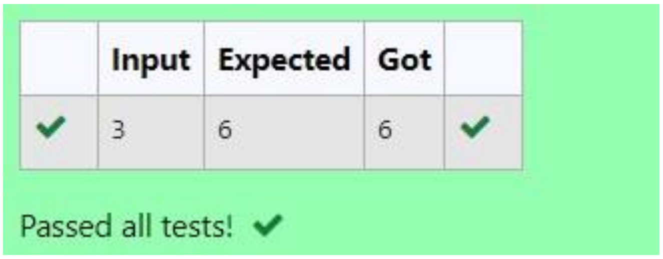
Step 1: Initialize x and sum=0 as integer data
type. Step 2: Take an input for x from the user.

Step 3: In a for loop, i=1, i<=x, Calculate sum+=i Step
4: Display sum.

PROGRAM:

```
#include<stdi
o.h> int main()

{

   int              x,sum=0;
   scanf("%d",   &x);   for
   (int i=1;i<=x;i++)

   {

      sum+=i;

   }

   printf("%d",sum);

}
```

OUTPUT:



	Input	Expected	Got	
✓	3	6	6	✓

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

PROGRAM

program to find the

12:

AIM: Write a C

ALGORITHM:

Nth term in the fibonacci series.

Step 1: Initialize n, f0=0, f1=1, f2 and z=0, o=1 as integer data type. Step 2: Take an input for n.

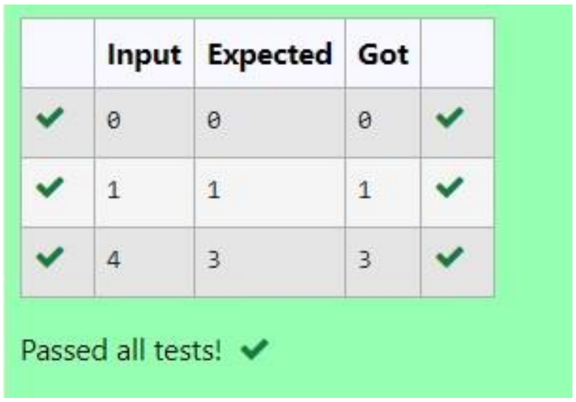
Step 3: Check if n==0, →
Display z Else if n==1 →
Display 0

Else calculate f2=f1+f0, f0=f1 and f1=f2 within a for loop Step 4: Display f2.

PROGRAM:

```
#include<stdi
o.h> int main()

{

   int       n,f0=0,f1=1,f2,z=0,o=1;

   scanf("%d",&n);        if(n==0)

   printf("%d",z);

   else if(n==1) printf("%d",o);
   else{

      for(int
        i=1;i<n;i++){ f2=f1+f0;

        f0=f
        1;

        f1=f
        2;

      }printf("%d",f2);

    }}
```

OUTPUT:



	Input	Expected	Got	
✓	0	0	0	✓
✓	1	1	1	✓
✓	4	3	3	✓

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

13:

AIM: Write a C powers of integers.

ALGORITHM:

```
Step 1: Initialize y, x and p as integers.
Step 2: Take an input from the user for x and y.
Step 3: calculate p as p=pow(x,y) and display p.
```

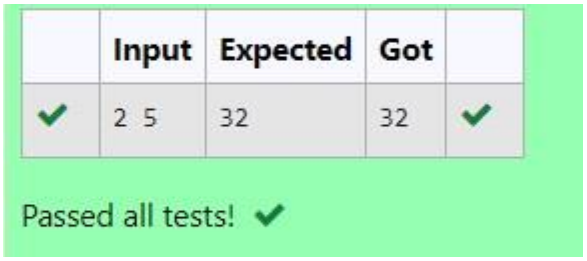
PROGRAM

program to find the

PROGRAM:

```
#include<stdi o.h>
#include<mat
h.h> int main()
{
    int y,x,p;
    scanf("%d%d",&x, &y);
    p=pow(x,y);
    printf("%d",p);
}
```

OUTPUT:



	Input	Expected	Got	
✓	2 5	32	32	✓

Passed all tests! ✓

RESULT: Thus, the program is executed successfully.

14:

AIM: Write a C

ALGORITHM:

whether the integer is prime or not.

Step 1: Initialize m as integer. Step 2: Take an input for m.

Step 3: Check if m%2!=0 and m%3!=0 and m%5!=0 →
Display prime Else → display not prime.

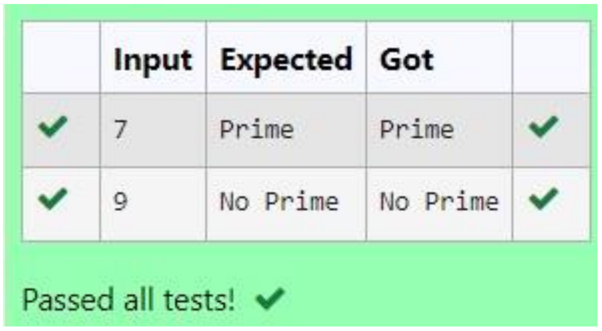
PROGRAM:

```
#include<stdi
o.h> int main()
{
   int m; scanf("%d",& m); if (m%2!=0 &&
   m%3!=0 && m%5!=0)
   {
      printf("Prime");
   }
   else
   {
      printf("No Prime");
   }
}
```

PROGRAM

program to find

OUTPUT:



	Input	Expected	Got	
✓	7	Prime	Prime	✓
✓	9	No Prime	No Prime	✓

Passed all tests! ✓

RESULT: **Thus, the program is executed successfully.**

15:

AIM: Write a C reverse of integer

ALGORITHM:

Step 1: Initialize m, rev=0 and rem as integers. Step 2: Take an input for m

Step 3: While m!=0 → rem=n%10 rev=rev*10+rem and m/=10

Step 4: Display rev

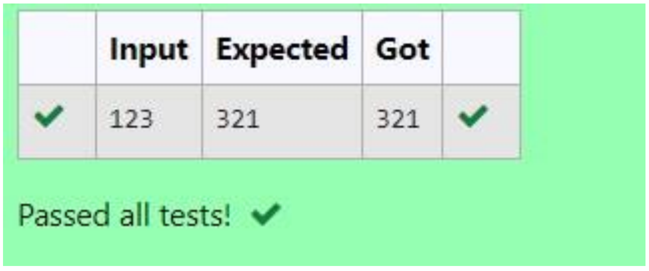
PROGRAM:

```
#include<stdi
o.h> int main()
{ int
   m,rev=0,rem;
   scanf("%d",& m);
   while(m!=0)
   {
      rem=m%10;
      rev=rev*10+r em;
      m/=10;
   }
   printf("%d",rev);
}
```

OUTPUT:

PROGRAM

program to find



	Input	Expected	Got	
✓	123	321	321	✓

Passed all tests! ✓

<u>RESULT:</u> Thus, the program is executed successfully.

WEEK 2: FINDING TIME COMPLEXITY

PROGRAM 1:

AIM:

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;

        }

}
```

ALGORITHM:

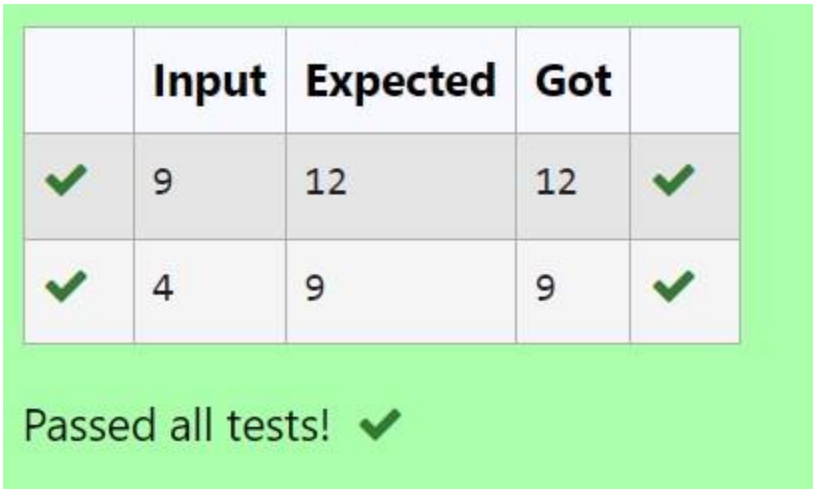
Step 1: Initiliaze a counter variable c=0 Step 2: Place c++ after each statement Step 3: Display c

PROGRAM:

```
#include<stdi

o.h> void func(int

n)
```

```
{ int
   c=1;
   int
   i=1;
   c+=1;
   int s=1;
   c+=1;
   while(s<=n)
   {
      c+=
      1; i+=1
      ;
      c+=
      1;
      s+=i
      ;
      c+=
      1;
   }
   printf("%d",c);
}

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

OUTPUT:



	Input	Expected	Got	
✓	9	12	12	✓
✓	4	9	9	✓

Passed all tests! ✓

RESULT: Thus the program is executed successfully.

PROGRAM 2:

AIM:

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;

       }

     }


  }

 }
```

ALGORITHM:

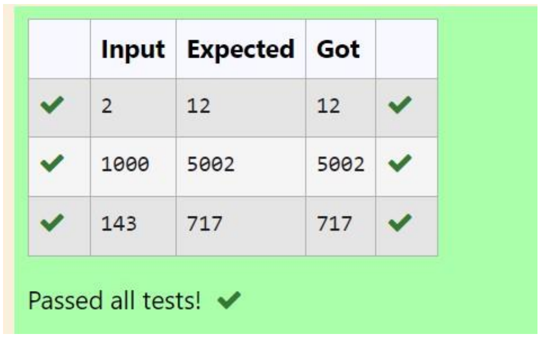
Step 1: initialize a counter variable c=0

Step 2: Place c++ after each iteration of a loop and declaration of a statement. Step 3: Display c

PROGRAM:

```
#include<stdi o.h>
int c=0;


void func(int n)

{

    if (n==1)

    {

        c++; printf("*"); }

    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! ✓

RESULT: Thus the program executed successfully.

PROGRAM 3:

AIM:

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);
           }
            }
 }
```

ALGORITHM:

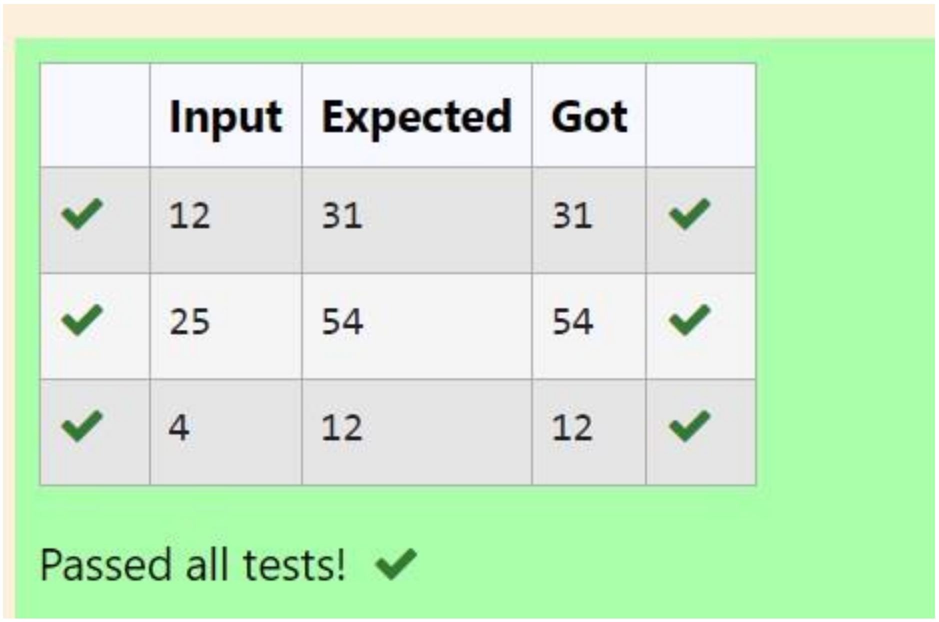
Step 1: initialize a variable c=0

Step 2: Place c++ after each iteration of a loop.
Step 3: display c

PROGRAM:

```
#include<stdi

o.h>  void  fac(int

n) {
    int c=0; for(int
    i=1;i<=n;++i)
    {
        c++; if
        (n%i==0)
        {
            c++;
              //printf("%d ",i);
        }
        c++;
    }
    c++; printf("%d",c);
}
int main()
{ int x;
    scanf("%d",
    &x); fac(x);
}
```

OUTPUT:



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

Passed all tests! ✓

RESULT: Thus the program is executed successfully.

PROGRAM 4:

AIM:

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++;

}
```

ALGORITHM:

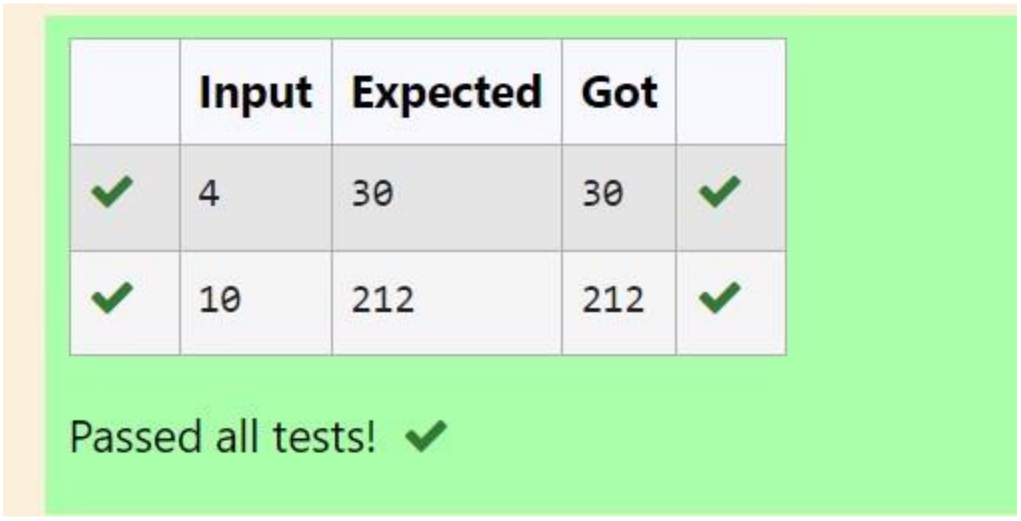
Step 1: Initialize a counter variable
c=0 Step 2: Place c++ after every
loop Step 3: display c

PROGRAM:

```
#include<stdio

.h>            void

function(int n) {


   int c=0; int

   count=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 x;
    scanf("%d", &x);
    function(x);
}
```

OUTPUT:



	Input	Expected	Got	
✓	4	30	30	✓
✓	10	212	212	✓

Passed all tests! ✓

RESULT: Thus the program is executed successfully.

PROGRAM 5:

AIM:

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);

}
```

ALGORITHM:

Step 1: Initialise the counter variable c=0

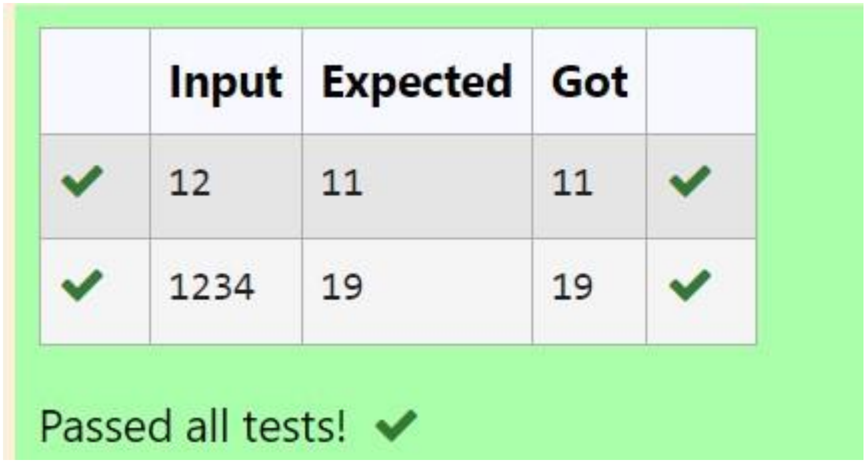
Step 2: After every iteration of a loop place a c++

Step 3: Display c

PROGRAM:

```
int count=0; void

reverse(int n)

{

   int rev = 0,

   remainder;

   count++; while

   (n != 0)

    {
        count++;
        remainder  =  n  %
        10; count++;
        rev = rev * 10 + remainder;
        count++;
        n/= 10; count+
        +;
    }
    count++;
    //print(rev);
    count++;
}
int main()
{ int n;
    scanf("%d",&n
    ); reverse(n);
    printf("%d",co unt);
}
```

OUTPUT:



	Input	Expected	Got	
✓	12	11	11	✓
✓	1234	19	19	✓

Passed all tests! ✓

RESULT: Thus the program executed successfully.

WEEK 3: GREEDY ALGORITHMS

PROGRAM 1:

AIM: 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.

ALGORITHM:

Step 1: Initialize all the variables required

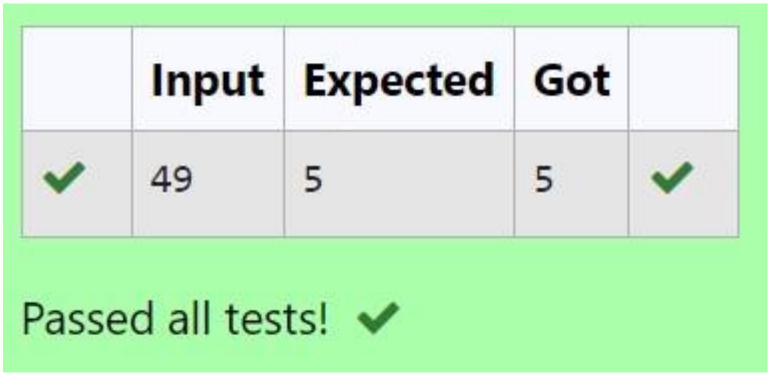
Step 2: Define an array den[] and then take an input

Step 3: Iterate through the array and calculate c+=d/den[i] if den[i]<d Step 4: Display C

PROGRAM:

```
#include<stdio

.h> int main()

{

   int d,c=0; scanf("%d",&d);


   int den[]={1000,500,100,50,20,10,5,2,1};

   int i=0;
   while(den[i] >d)
   {
      i++;
   }
   while(d!=0)
   {
      if (den[i]<d)
      {
         c+=d/den[i]; d=d%den[i];
      }
      i++;
   }
   printf("%d",c);
}
```

OUTPUT:



	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

RESULT: Thus the program executed successfully.

PROGRAM 2:

AIM: 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.

ALGORITHM:

Step 1: Input the size of the first array g[] and its elements.
Step 2: Input the size of the second array s[] and its
elements. Step 3: Compare each element of g[] with the
elements of s[]. Step 4:
Output the result.

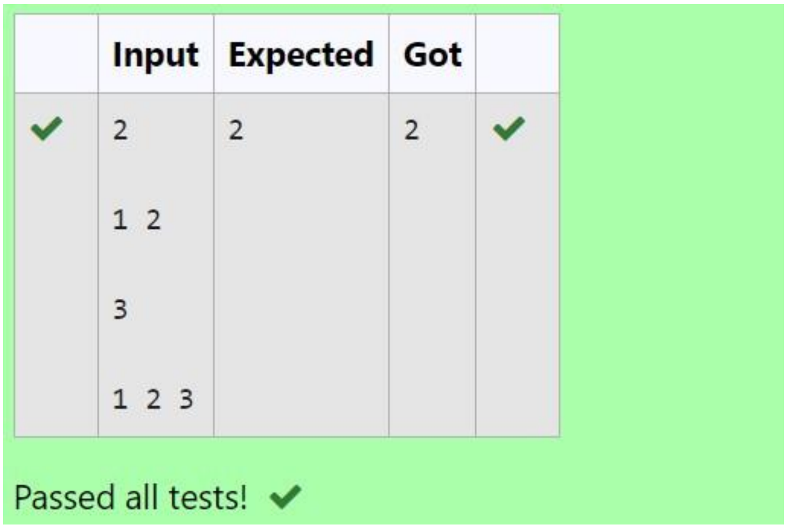
PROGRAM:

```
#include<stdi

o.h> int main()

{ int n;

   scanf("%d",
   &n); int g[n]; for (int

   i=0;i<n;i++)

   {

      scanf("%d",&g[i]);

   }

   int c,r=0; scanf("%d",
   &c); int s[c]; for(int

   j=0;j<c;j++)

   {

         scanf("%d",&s[j]);

   }

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

```
        {
              for (int j=0;j<c;j++)
          { if (s[j]>g[i])
              {
                  r++;
```

```
                break;
            }
        }
    }
    printf("%d",r);
}
```

OUTPUT:



	Input	Expected	Got	
✓	2 1 2 3 1 2 3	2	2	✓

Passed all tests! ✓

RESULT: Thus the program was executed successfully.

PROGRAM 3:

AIM: A person needs to eat burgers. Each burger contains a count of calories. 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 3i * 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 (30 * 1) + (31 * 3) + (32 * 2) = 1 + 9 + 18 = 28.

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

ALGORITHM:

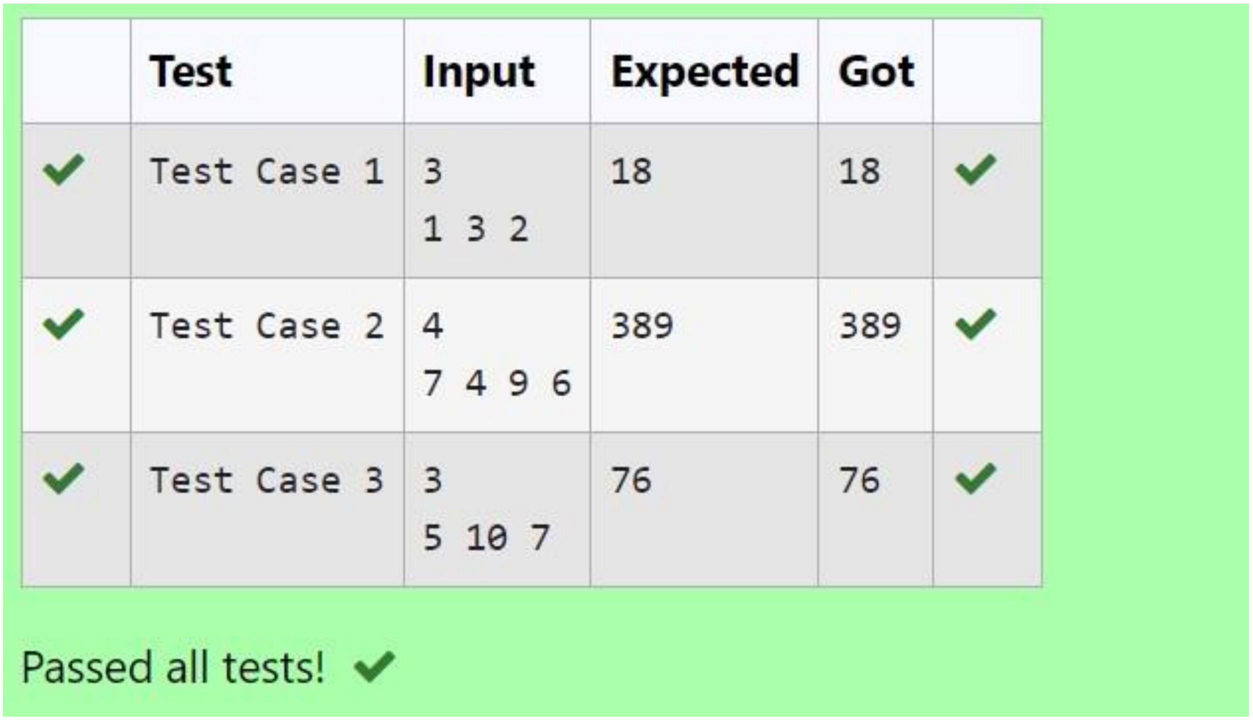
Step 1: Input the size of the array a[] and its elements. Step
2: Sort the array in descending order.

Step 3: Calculate the sum with weighted powers. Step 4:
Output the result.

PROGRAM:

```
#include<stdio
.h>
#include<mat h.h>
#include<stdli b.h>


int compare(const void* a, const void* b)

{

   return (*(int*)b-*(int*)a);

}


int main()

{

   int n,sum=0; scanf("%d",
   &n); int a[n];

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

   {

      scanf("%d",&a[i]);

   }

   qsort(a,n,sizeof(int),compare);
for(int i=0;i<n;i++)
   {
      sum+=pow(n,i)*a[i];
   }
   printf("%d",sum);
}
```

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! ✓

RESULT: Thus the program was executed successfully.

PROGRAM 4:

AIM: 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).

ALGORITHM:

Step 1: Input the size of the array a[] and its elements. Step 2: Sort the array a[] in ascending order.

Step 3: Calculate the weighted sum.

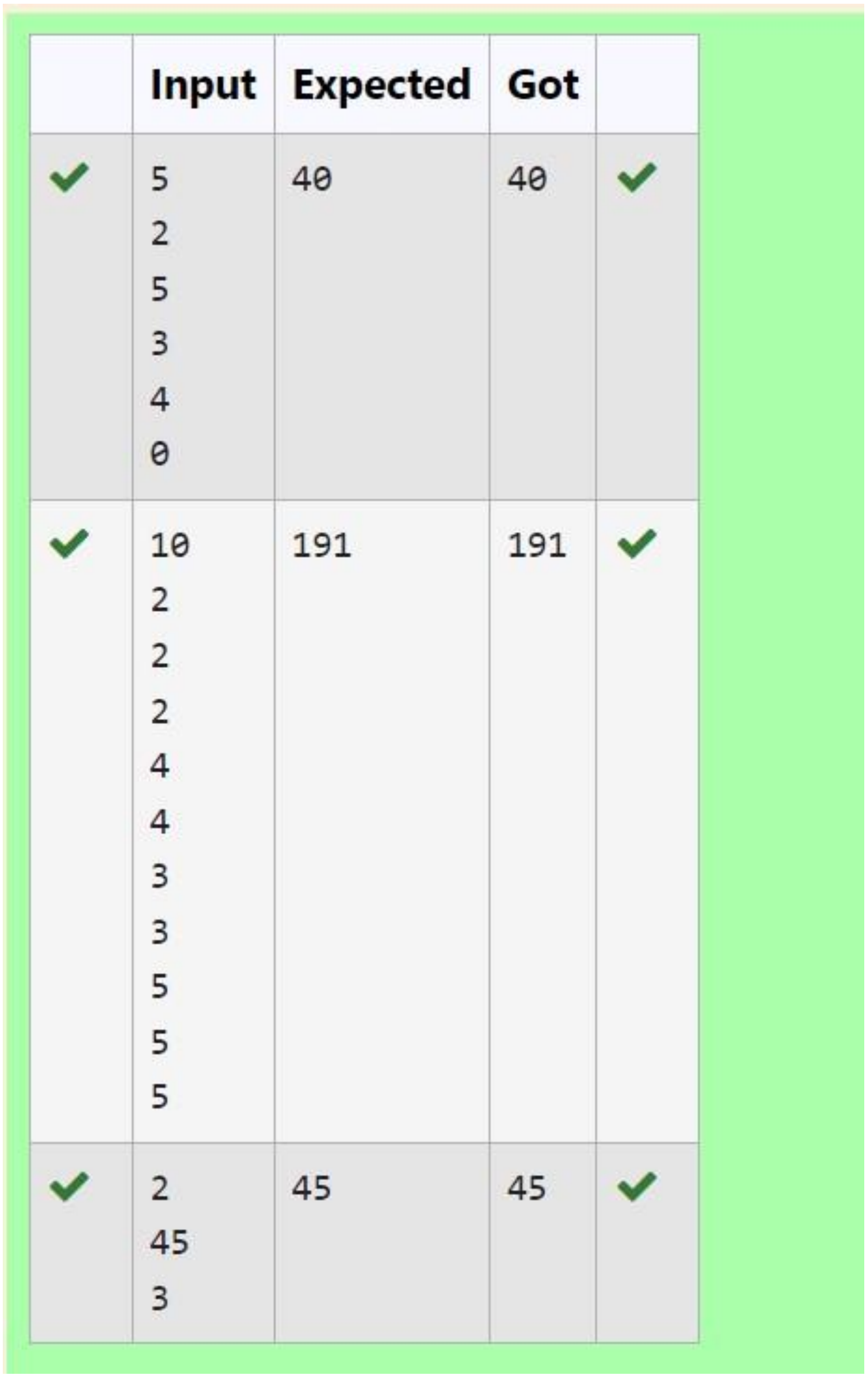
Step 4: Output the result.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>


int compare(const void* a,const void* b)

{

   return (*(int*)a-*(int*)b);

}


int main()

{

   int             n,sum=0;
   scanf("%d",  &n);  int
   a[n];      for      (int
   i=0;i<n;i++)

   {

      scanf("%d",&a[i]);

   }
```

```
    qsort(a,n,sizeof(int),compare);
 for (int j=0;j<n;j++)
    {
       sum+=a[j]*j;
    }
    printf("%d",sum);
 }
```

OUTPUT:



	Input	Expected	Got	
✓	5 2 5 3 4 0	40	40	✓
✓	10 2 2 2 4 4 3 3 5 5 5	191	191	✓
✓	2 45 3	45	45	✓

RESULT: Thus the program executed successfully.

PROGRAM 5:

AIM: 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.

ALGORITHM:

Step 1: Input the size of the arrays and the elements of both arrays a[] and b[]. Step 2: Sort array a[] in descending order and array b[] in ascending order.

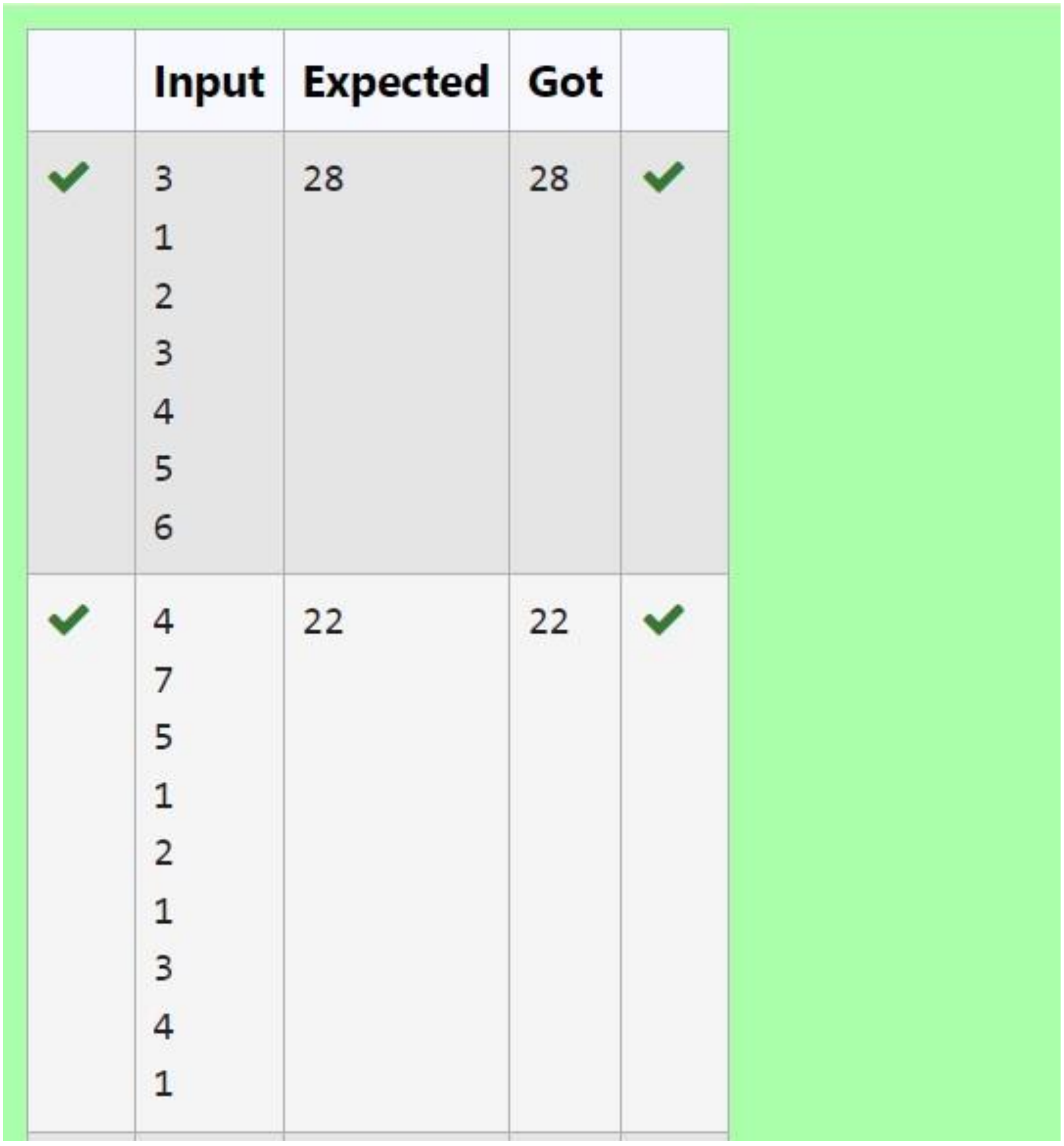
Step 3: Calculate the sum of products.
Step 4: Output the result.

PROGRAM:

```
#include<stdio
.h>
#include<stdli b.h>


int compare(const void* a,const void* b)

{

    return (*(int*)a-*(int*)b);

}


int compare1(const void* a,const void* b)

{

    return (*(int*)b-*(int*)a);

}
```

```
int main()

{

   int              n,sum=0;

   scanf("%d",  &n);  int

   a[n],b[n];    for    (int

   i=0;i<n;i++)

   {

      scanf("%d",&a[i]);

   }

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

   {

      scanf("%d",&b[j]);

   }


   qsort(b,n,sizeof(int),compare); qsort(a,n,sizeof(int),compare1);

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

   {

      sum+=a[k]*b[k];

   }

   printf("%d",sum);

}
```

OUTPUT:



	Input	Expected	Got	
✓	3 1 2 3 4 5 6	28	28	✓
✓	4 7 5 1 2 1 3 4 1	22	22	✓

<u>RESULT:</u> **Thus the program was executed successfully.**

<u>WEEK 4: DIVIDE AND CONQUER</u>

<u>PROGRAM 1:</u>

<u>AIM:</u> 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.

ALGORITHM:

Step 1: Input the size of the array and the elements.

Step 2: Define the recursive divide function to find the first occurrence of 1. Step 3: Call the divide function and compute the result.

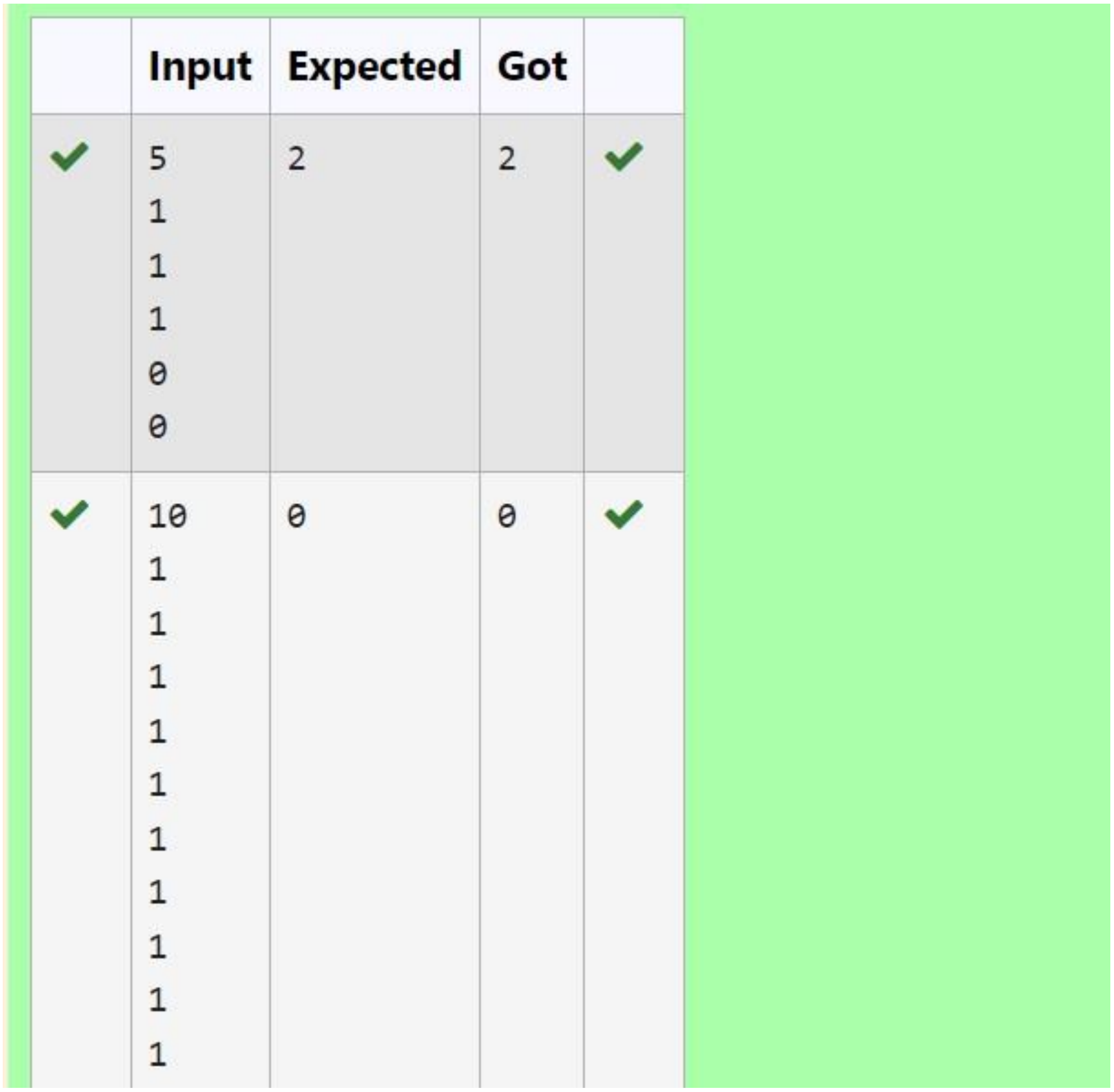
Step 4: Output the result.

PROGRAM:

```
#include   <stdio.h>   int   divide(int
[],int,int);  int  divide(int  a[],int  left,int
right)
{
   int              mid=0;
   mid=left+(rightleft)/
   2; if (a[0]==0) return
   0;
   else               if
      (a[right1]==1)
      return right;
   if        ((a[mid]==0)        &&
      (a[mid1]==0))          return
      divide(a,0,mid);
   else if
      (a[mid]==0)   return   mid;   else
   return divide(a,mid+1,right);
}

int main()
```

```
{ int n;
    scanf("%d",
    &n); int arr[n]; for (int
    i=0;i<n;i++)
    {
            scanf("%d",&arr[i]);
    } int
   zero=divide(arr,0,n );
   printf("%d",nzero);


}
```

OUTPUT:



	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1 1 1	0	0	✓

RESULT: Thus the program is executed successfully.

PROGRAM 2:

AIM: 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.

ALGORITHM:

Step 1: Input the size of the array and its elements.

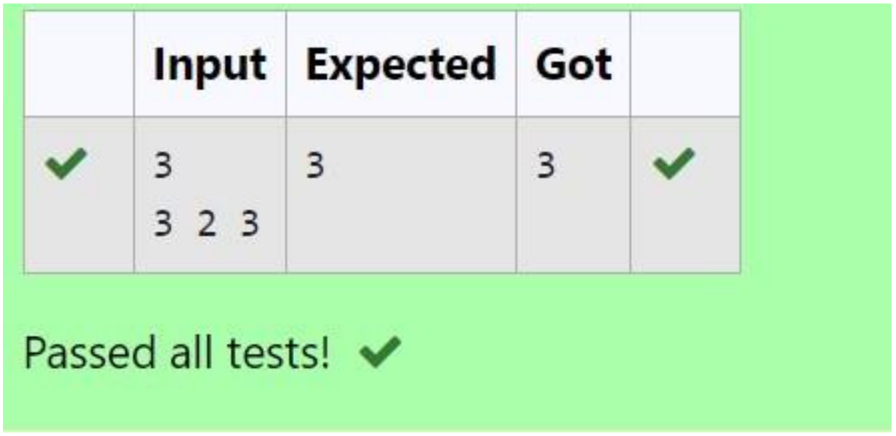
Step 2: Define the recursive function Count to count occurrences of a specific element (key). Step 3: Find the majority element and check if its count exceeds half the array size.

Step 4: Handle edge cases where k is not the majority element. Step 5: Display the output

PROGRAM:

```
#include<stdio.h>


#include
<stdio.h> int mid=0,c=0;

int Count(int [],int,int,int); int Count(int a[],int

left,int right,int key)

{

   int mid=left+(right-
   left)/2; if
   (a[mid]==key) c++;

   else

   {

        Count(a,left,mid,key);
          Count(a,mid+1,right,key);

   }

   return c;

}

int main()

{ int n;

   scanf("%d",
   &n); int arr[n];

   for (int i=0;i<n;i++)
       scanf("%d",&arr
       [i]);

   int k=arr[0]; if

   (Count(arr,0,n,k)>n/2)

   printf("%d"

   ,k); else

   { for (int

        i=0;i<n/2;i++) if (arr[i]!=k)

         {

             printf("%d"
             ,k); break;

         }

   }


}
```

OUTPUT:



	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

RESULT: Thus the program is executed successfully.

PROGRAM 3:

AIM: Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

ALGORITHM:

Step 1: Input the size of the array and its elements.

Step 2: Define the search function to find the largest element smaller than or equal to x. Step 3: Call the search function and get the result.

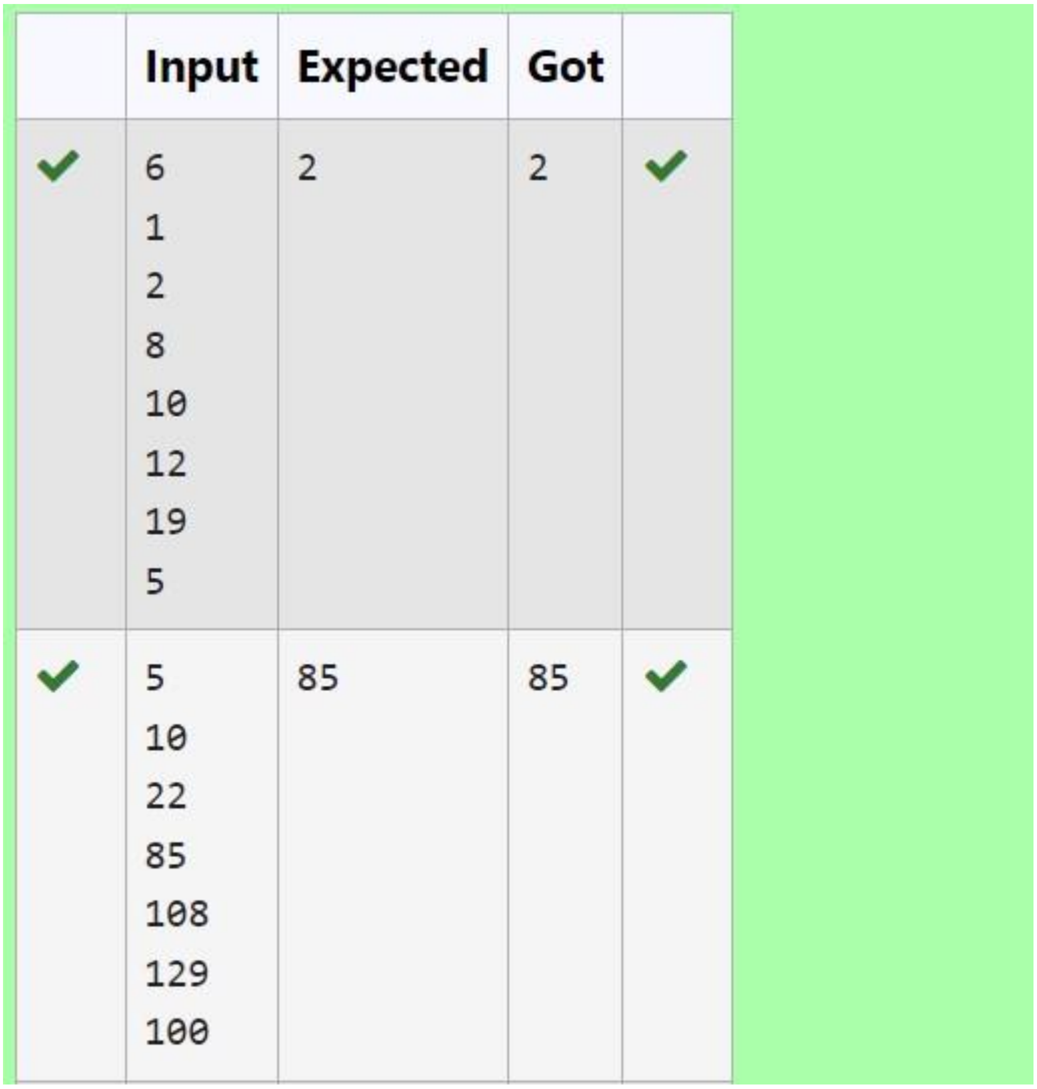
Step 4: Output the result.

PROGRAM:

```
#include<stdio.h>

int search(int arr[], int n, int x)

{

   if (x>=arr[n1])
      return n-1;

   if
      (x<arr[ 0])
      return -
      1;


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

```
        (arr[i]>x) return
            arr[i-1];


    return -1;
}


int main()
{ int n;
    scanf("%d",
    &n); int a[n]; for (int
    i=0;i<n;i++)
    {
            scanf("%d",&a[i]);
    }
    int x;
    scanf("%d", &x);
    int res=search(a, n,
    x); if (res!=-1)
    printf("%d",res);
}
```

OUTPUT:



	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 108 129 100	85	85	✓

<u>RESULT:</u> **Thus the program is executed successfully.**

PROGRAM 4:

<u>AIM:</u> **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

ALGORITHM:

Step 1: Input the size of the array and its elements.

Step 2: Define the sum function to find two elements whose sum equals x. Step 3: Call the sum function to find the pair.

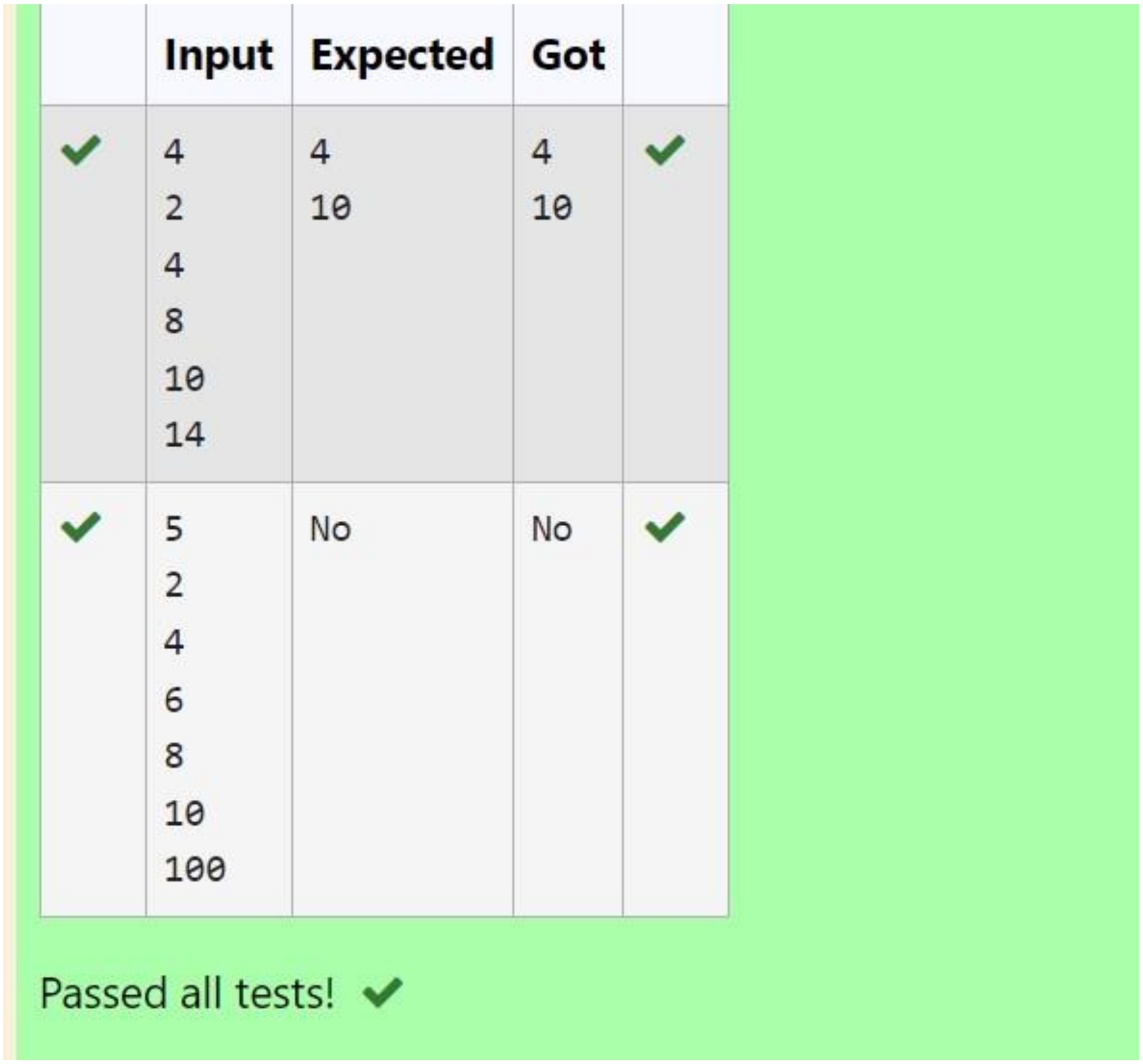
Step 4: Output the result.

PROGRAM:

```
#include<stdio.h>

void sum(int a[], int l, int r, int x)

{ if (l>=r)

   {

      printf("No\ n");
      return;

   }


   int ts=a[l]+a[r];


   if (ts==x)

   {

      printf("%d\n",a[l]); printf("%d\n",a[r]);

   }

   else if (ts<x)

   {

        sum(a,l+1,r,x);

   }

   else

   {
```

```
            sum(a,l,r-1,x);
        }
    }

    int main()
    {
        int n,x; scanf("%d",
        &n); int arr[n]; for (int
        i=0;i<n;i++)
        {
            scanf("%d",&arr[i]);
        }
        scanf("%d",&x); sum(arr,0,n-1,x);
    }
```

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! ✓

RESULT: **Thus the program is executed successfully.**
PROGRAM 5:

AIM: **Write a Program to Implement the Quick Sort Algorithm**

ALGORITHM:

Step 1: Input the array size and elements. Step
2: Define the swap function.

Step 3: Define the partition function. Step 4: Define the quicksort function.

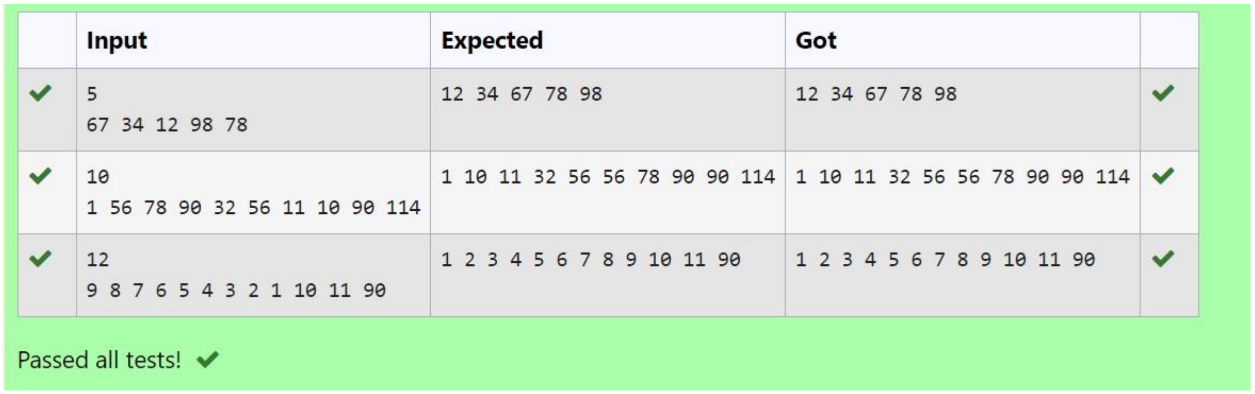
Step 5: Call the quicksort function in the main() function.

PROGRAM:

```
#include
<stdio.h>
#include
<stdlib.h>


void swap(int *p1, int *p2)

{

   int   temp;
   temp      =
   *p1;

   *p1 = *p2;

   *p2 = temp;

}


int partition(int a[], int low, int high)

{

   int p = a[high]; int i = low - 1;
   for (int j = low; j < high; j++)

   { if (a[j] < p)

      {

         i++; swap(&a[i],

         &a[j]);

      }
```

```
    }
    swap(&a[i + 1],
    &a[high]); return (i + 1);
}

void quicksort(int a[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(a, low, high);
        quicksort(a,   low,   pi   -   1);
        quicksort(a, pi + 1, high);
    }
}

int main()
{ int n; scanf("%d", &n); int
    a[n]; for (int i = 0; i < n;
    i++)
    {
        scanf("%d", &a[i]);
    }
    quicksort(a, 0, n -
    1); for (int i = 0; i < n;
    i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n ");
    return
    0;

}
```

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! ✓

RESULT: **Thus the program is executed successfully.**

WEEK 5: DYNAMIC PROGRAMMING

PROGRAM 1:

AIM: 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.

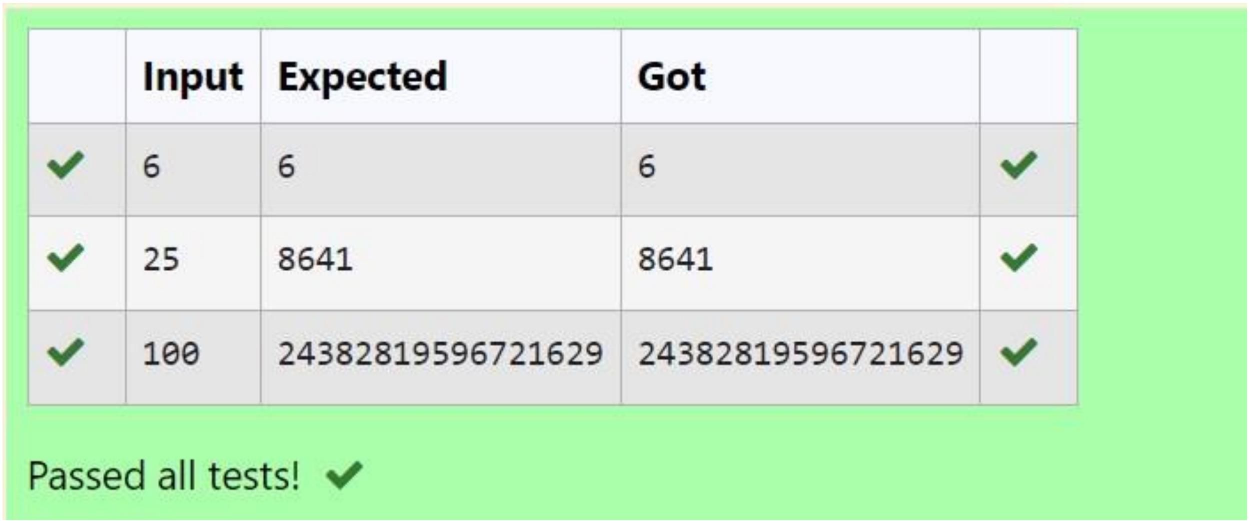
ALGORITHM:

```
Step 1: Start
Step 2: Declare n and read input value
Step 3: Create an array dp of size n + 1, initialize dp[0] to 1,
and set all other elements to 0
Step 4: Iterate from 1 to n, updating dp[i] by adding dp[i - 1]. If i is
greater than or equal to 3, also add dp[i - 3] to dp[i]
Step 5: Print the value dp[n]
Step 6: End
```

PROGRAM:

```
#include<stdi
o.h> int main()
{
   int            n;
   scanf("%d",
   &n);      long
```

```
    dp[n+1];
    dp[0] = 1;

    for (int i = 1; i <= n; i++) {
    dp[i] = 0;

    } for (int i = 1; i <= n;

    i++) { dp[i] += dp[i -

    1]; if (i >= 3) { dp[i] +=

    dp[i - 3];

        }

    }

    printf("%ld\n",     dp[n]);
    return 0;

}
```



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

Passed all tests! ✓

<u>RESULT:</u> Thus the program is executed successfully.

PROGRAM 2:

AIM: 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.

ALGORITHM:

Step 1: Start

Step 2: Declare n, read input value, and create a 2D array board of size n x n to store its values Step 3: Initialize dp[0][0] with board[0][0], populate the first row and column of dp by accumulating values from board

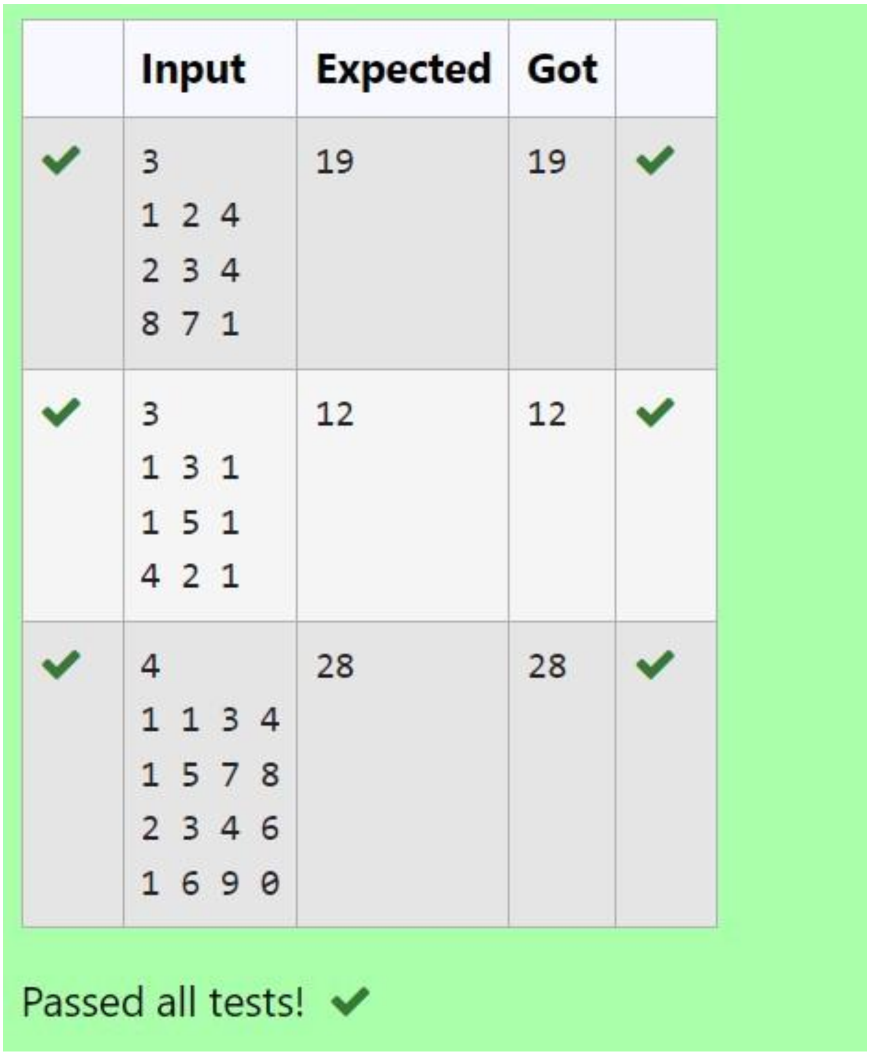
Step 4: Iterate through the remaining cells of the dp array, updating each cell with the maximum path sum from either the top or left cell, and board[i][j] Step 5: Print the value dp[n-1][n-1]

PROGRAM:

```
#include<stdio. h> int

max(int   a,int   b)   {

return(a>b) ? a:b;

}


int  maxMonetaryPath(int  n,int  board[n][n]){  int
   dp[n][n]; 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[i1][0]+board[i][0];

    } for

    (int

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

                    dp[i][j]=board[i][j]+max(dp[i-1][j],dp[i][j-1]);

        }

    }

    return dp[n-1][n-1];

}


int main(){

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

        }

    } int

    result=maxMonetaryPath(n,board
    ); printf("%d\n",result);

}
```

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! ✓

RESULT: Thus the program is executed successfully.

PROGRAM 3:

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

ALGORITHM:

Step 1: Start

Step 2: Declare s1 and s2 as character arrays and read the input values

Step 3: Calculate the lengths of s1 and s2, and create a 2D array dp of size (len1 + 1) x (len2 + 1)

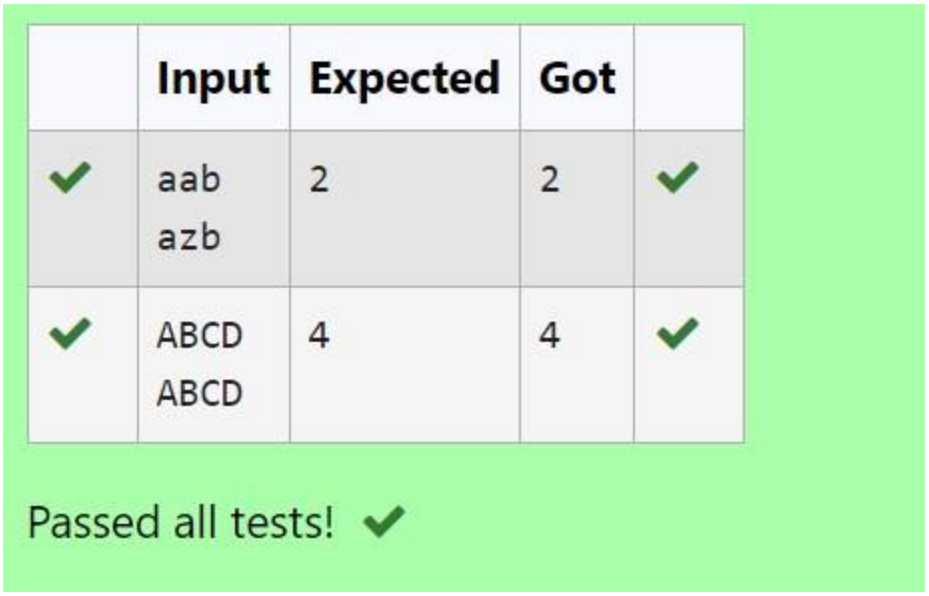
Step 4: Initialize the first row and column of dp to 0, then iterate through the arrays to fill dp by comparing characters of s1 and s2 and taking the maximum of the adjacent values

Step 5: Print dp[len1][len2]

PROGRAM:

```
#include<stdio
.h>
#include<strin g.h>


int main()

{

   char s1[10],s2[10]; scanf("%s",s1);

   scanf("%s",s2);


   int len1=strlen(s1)
   ; int
   len2=strlen(s2)
   ;

   int dp[len1 + 1][len2 + 1];
   for(int i=0;i<=len1;i++)

   {

      for(int j=0;j<=len2;j++)

      {

         if(i==0||j==0)

         {

            dp[i][j]=0;
            }
    else if(s1[i-1]==s2[j-1]){ dp[i][j]=dp[i-1][j-1]+1;
            }
            else{ if(dp[i][j-1]>dp[i-1][j])
                dp[i][j]=dp[i][j-
                1]; else dp[i][j]=dp[i-1][j];

                }
            }
        }
            printf("%d",dp[len1][len2]);
 }
```

OUTPUT:



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

Passed all tests! ✓

RESULT: Thus the program executes successfully.

PROGRAM 4:

AIM: Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

ALGORITHM:

Step 1: Start

Step 2: Declare n, read the input value, and create an array arr of size n to store its values
Step 3: Initialize a 1D array dp of size n with all elements set to 1, and a variable maxlen to 1 Step 4: Iterate through the array arr to fill dp by comparing

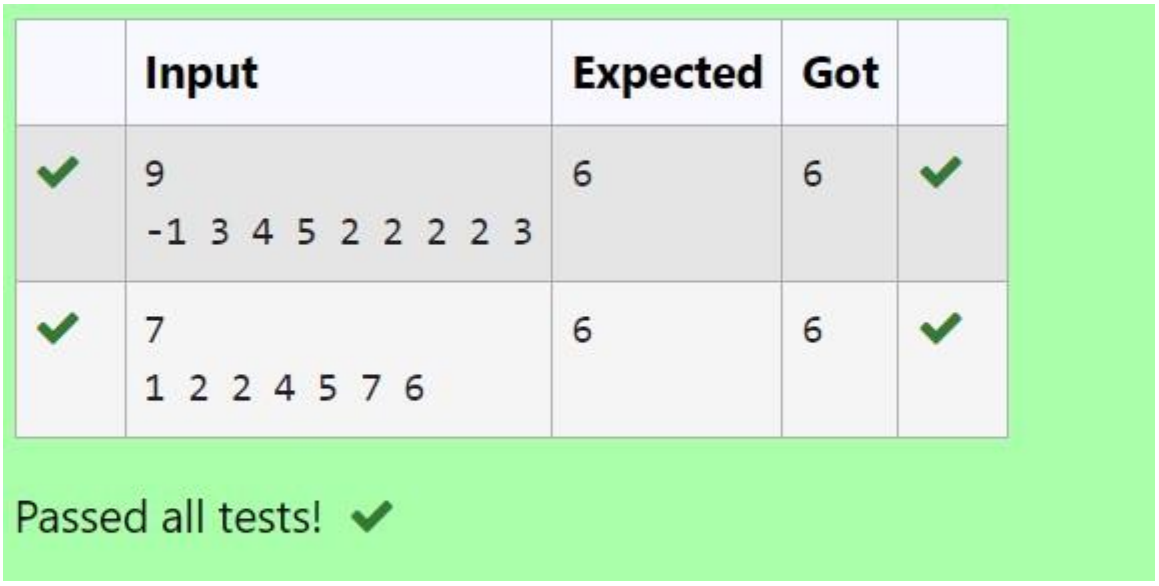
elements, updating dp[i] if arr[i] is greater than or equal to arr[j] and dp[i] < dp[j] + 1, also update maxlen.

Step 5: Print maxlen.

PROGRAM:

```
#include <stdio.h>


int subsequence(int
    arr[],int n){ int dp[n]; int

    maxlen=1;

    for (int i=0;i<n;i++){
        dp[i]=1;

    } for

    (int

        i=1;i<n;i++){

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

            if(arr[i]>=arr[j] &&
                    dp[i]<dp[j]+1){ dp[i]=dp[j]+1;

            }

        }

    if(maxlen<dp
        [i]){
        maxlen=dp
        [i];
        }
    }
    return maxlen;
}


int
    main()
    { int n;
    scanf("%d",
    &n); int arr[n];
    for (int
        i=0;i<n;i++){ scanf("%d",&arr[i
        ]);
    }
    int result=subsequence(arr,n); printf("%d",result);
}
```

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! ✓

RESULT: Thus the program was executed successfully.

WEEK 6: COMPETITIVE PROGRAMMING

PROGRAM 1:

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

ALGORITHM:

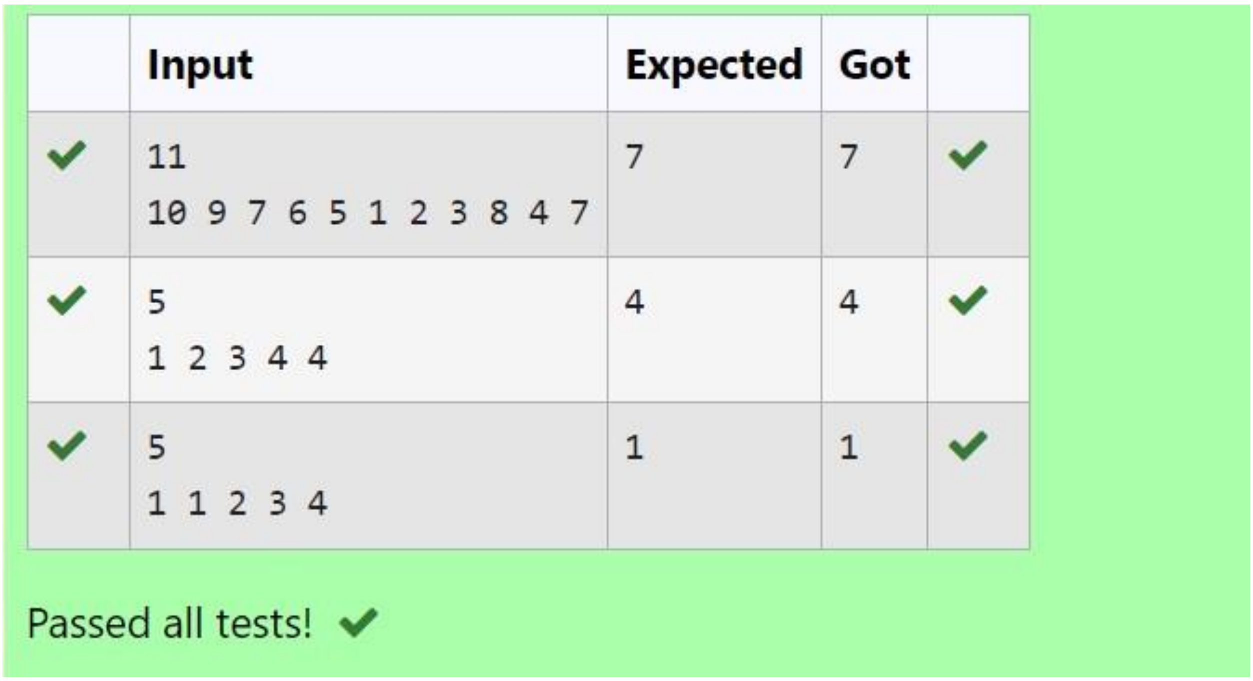
Step 1: Input the size of the array and the array elements. Step 2: Sort the array using QuickSort.

Step 3: Search for the first repeated element. Step 4: Output the result.

PROGRAM:

```
 #include<stdio
 .h>
 #include<stdli b.h>


 int compare(const void* a, const void* b)

 {

     return (*(int*)a-*(int*)b);

 }

in

t

m

ai

n(

)

 {

     int n,temp,p; scanf("%d",
     &n); int a[n]; for (int

     i=0;i<n;i++)

     {

         scanf("%d",&a[i]);

     }


     qsort(a,n,sizeof(int),compar e);
     for(int i=0;i<n;i++)

     {
        if (a[i]==a[i+1])
        {
            p=1; temp=a[i];
        }
    }
    if (p==1)
    {
        printf("%d",temp);
    }
}
```

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! ✓

RESULT: Thus the program executed successfully.

PROGRAM 2:

AIM; Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and declare an array arr of size n.

Step 3: Read the first value into t and assign it to arr[0].

Step 4: Iterate from index 1 to n-1, reading values into arr[i]. If the value of t matches arr[i], break the loop. Otherwise, update t to arr[i].

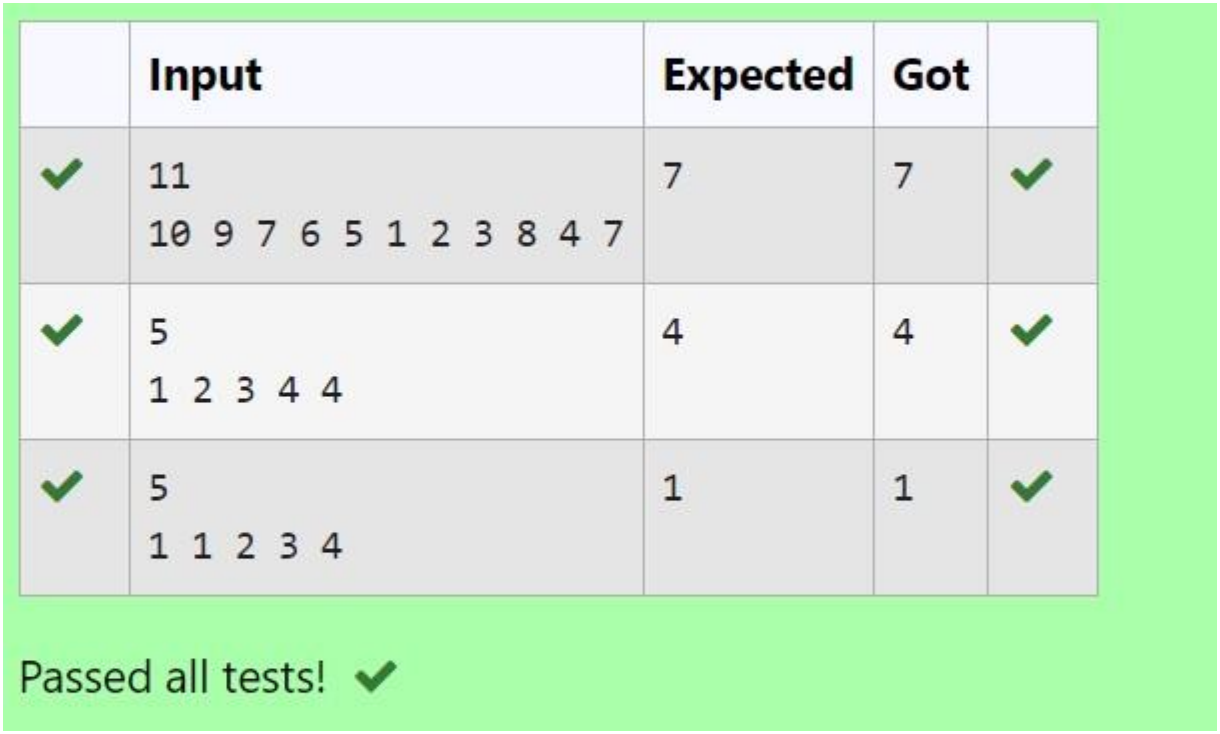
Step 5: After the loop, print the value of t.

Step 6: End

PROGRAM:

```
#include<stdi
o.h> int main()

{ int n,t;

   scanf("%d",
   &n); int arr[n];
   scanf("%d", &t);
   arr[0]=t; for(int
   i=1;i<n;i++){
   scanf("%d",&arr
      [i]); if(t==arr[i])

      break
      ; else t=arr[
      i];

   }

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

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! ✓

RESULT: **Thus the program executes successfully.**

PROGRAM 3:

AIM: **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.

ALGORITHM:

Step 1: Start

Step 2: Read the number of test cases, t

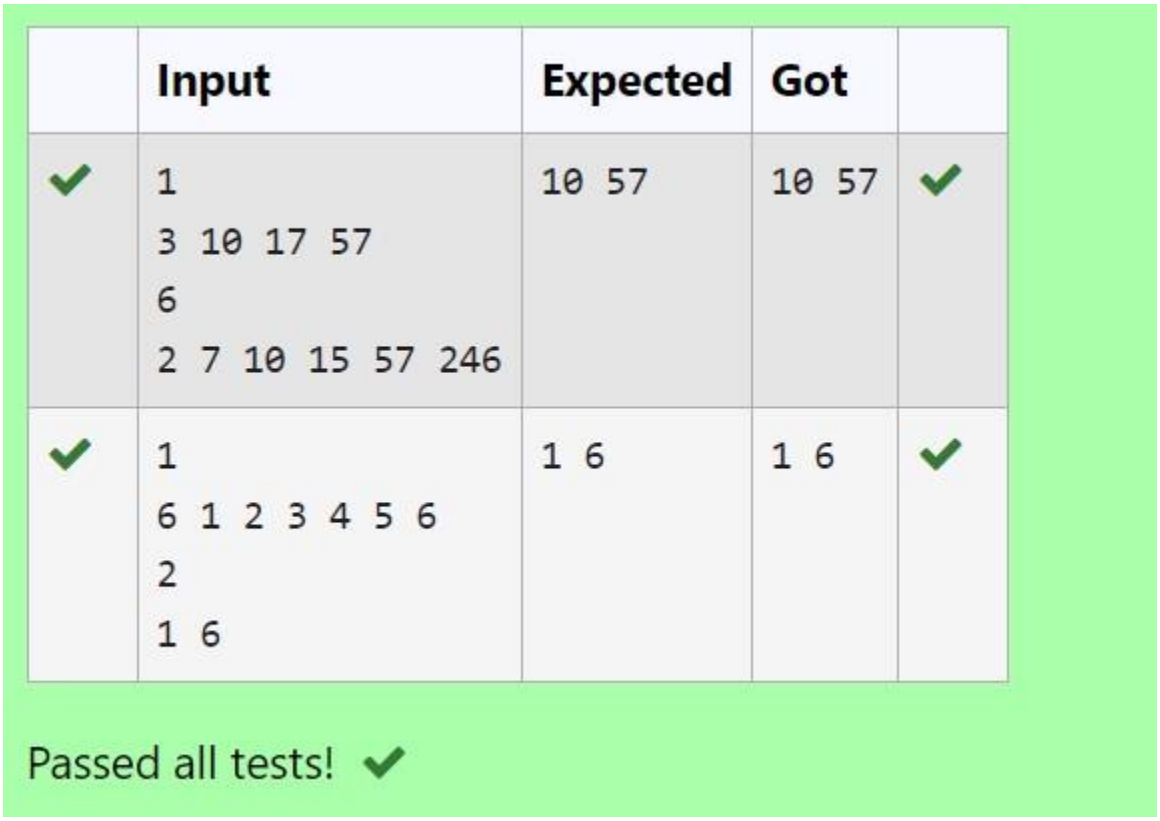
Step 3: For each test case, read the sizes n1 and n2 and the elements of the arrays arr1 and arr2

Step 4: For each element in arr1, check if it exists in arr2. If it does, print the element Step 5: End

PROGRAM:

```
#include <stdio.h>

void intersection(int arr1[],int n1,int arr2[],int n2)

{  for  (int  i=0;i<n1;i++){
    int    element=arr1[i];
    for (int j=0;j<n2;j++){

           if
               (arr2[j]==elemen

               t) { printf("%d
               ",element); break;

           }

       }

    } printf("\n");

} int

    main()
    { int t;

    scanf("%d",&t);
    while(t--){ int
    n1,n2;

        scanf("%d",&
        n1); int arr1[n1];

        for(int i=0;i<n1;i++){
                scanf("%d",&arr1[i]);

        }
    scanf("%d",&
    n2); int arr2[n2];
    for(int
        i=0;i<n2;i++){ scanf("%d",&arr
        2[i]);
    }
    intersection(arr1,n1,arr2,n2);
    }
}
```

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! ✓

RESULT: Thus the program executes successfully.

PROGRAM 4:

AIM:

ALGORITHM:

Step 1: Start

Step 2: Read the number of test cases, t

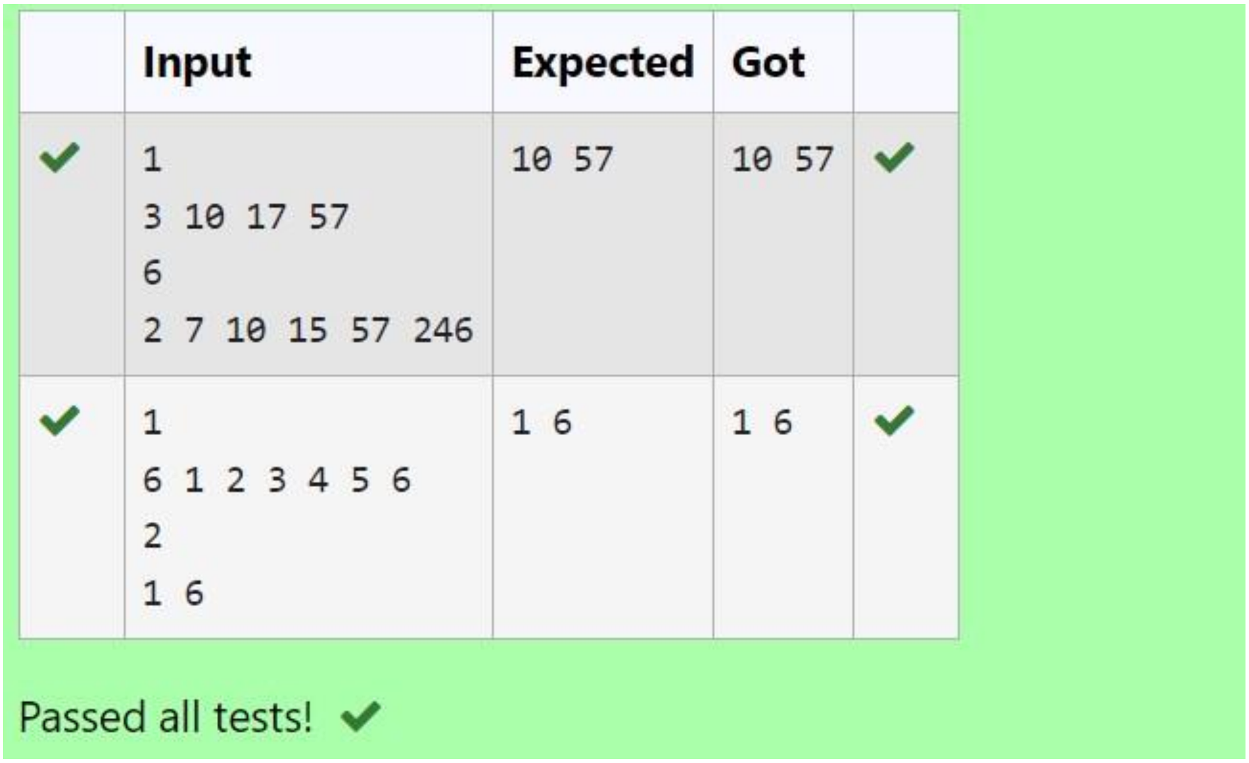
Step 3: For each test case, read the sizes n1 and n2, then read elements of the arrays arr1 and arr2

Step 4: Use two pointers to iterate through arr1 and arr2, printing the common elements
Step 5: End

PROGRAM:

```
#include <stdio.h>

void intersection(int arr1[], int n1, int arr2[], int n2)

{

  int   i=0,j=0;   while
  (i<n1  &&  j<n2){  if
  (arr1[i]<arr2[j])     {
  i++;

    }

    else if
      (arr2[j]<arr1[i]){ j++;

    }

    else{

      printf("%d

      ",arr1[i]); i++; j++;

    }
    } printf("\n");
}


int
    main()
    { int t;
    scanf("%d",&t);
    while (t-){ int
        n1,n2;
        scanf("%d", &n1);
        int arr1[n1];
        for (int i=0;i<n1;i++){
                scanf("%d",&arr1[i]);
        }
        scanf("%d",&
        n2); int arr2[n2];
        for (int
            i=0;i<n2;i++){ scanf("%d",
            &arr2[i]);
        }
    intersection(arr1,n1,arr2,n2);
    }
}
```

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! ✓

RESULT: Thus the program executes successfully.

PROGRAM 5:

AIM: 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.

ALGORITHM:

Step 1: Start

Step 2: Declare n, k and read the input values

Step 3: Create an array arr of size n and read its values

Step 4: Iterate through the array using nested loops to check if there is any pair whose difference is equal to k. If found, return 1.

If no such pair is found, return 0

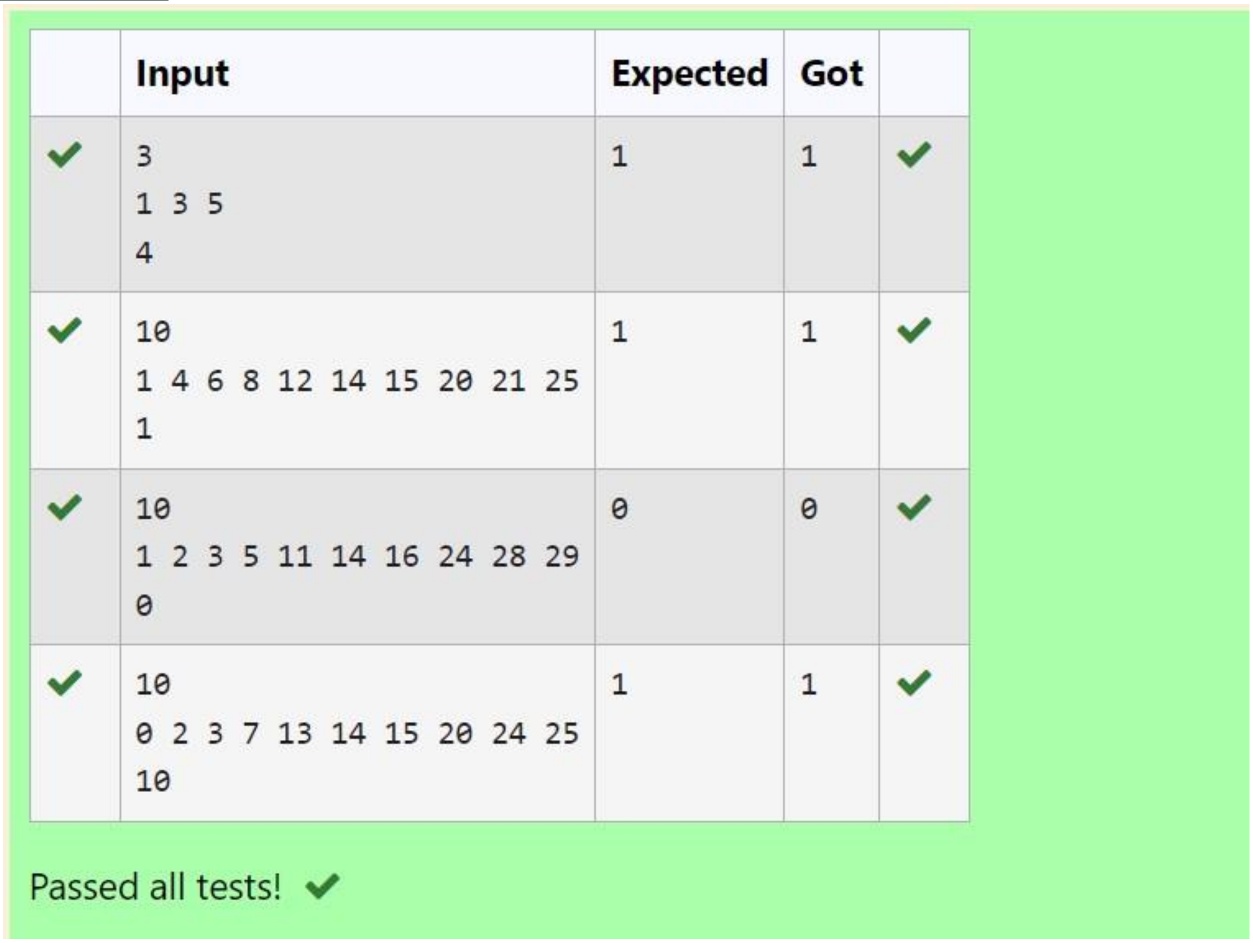
Step 5: Print the result and end the program

PROGRAM:

```
#include <stdio.h>


int checkpair(int arr[],int n,int k){
   for (int i=0;i<n;i++){

      for                  (int
         j=i+1;j<n;j++){
         if(arr[j]arr[i]==k
         ){ return 1;

         }

         else if(arr[j]arr[i]>k){
            break;

         }

      }

   }

   return 0;

}


int
   main()
   { int n, k;

   scanf("%d",
   &n); int arr[n];

   for (int i=0;i<n;i++)
      {
       scanf("%d",&arr
      [i]);

   }
```

```
    scanf("%d",&k);
    int result=checkpair(arr,n
    ,k); printf("%d\n",result);
}
```

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! ✓

RESULT: Thus the program executes successfully.

PROGRAM 6:

AIM: 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.

ALGORITHM:

Step 1: Start

Step 2: Declare n, k and read the input values

Step 3: Create an array arr of size n and read its values

Step 4: Use two pointers i and j to iterate through the array, checking if the difference between arr[j] and arr[i] is equal to k.

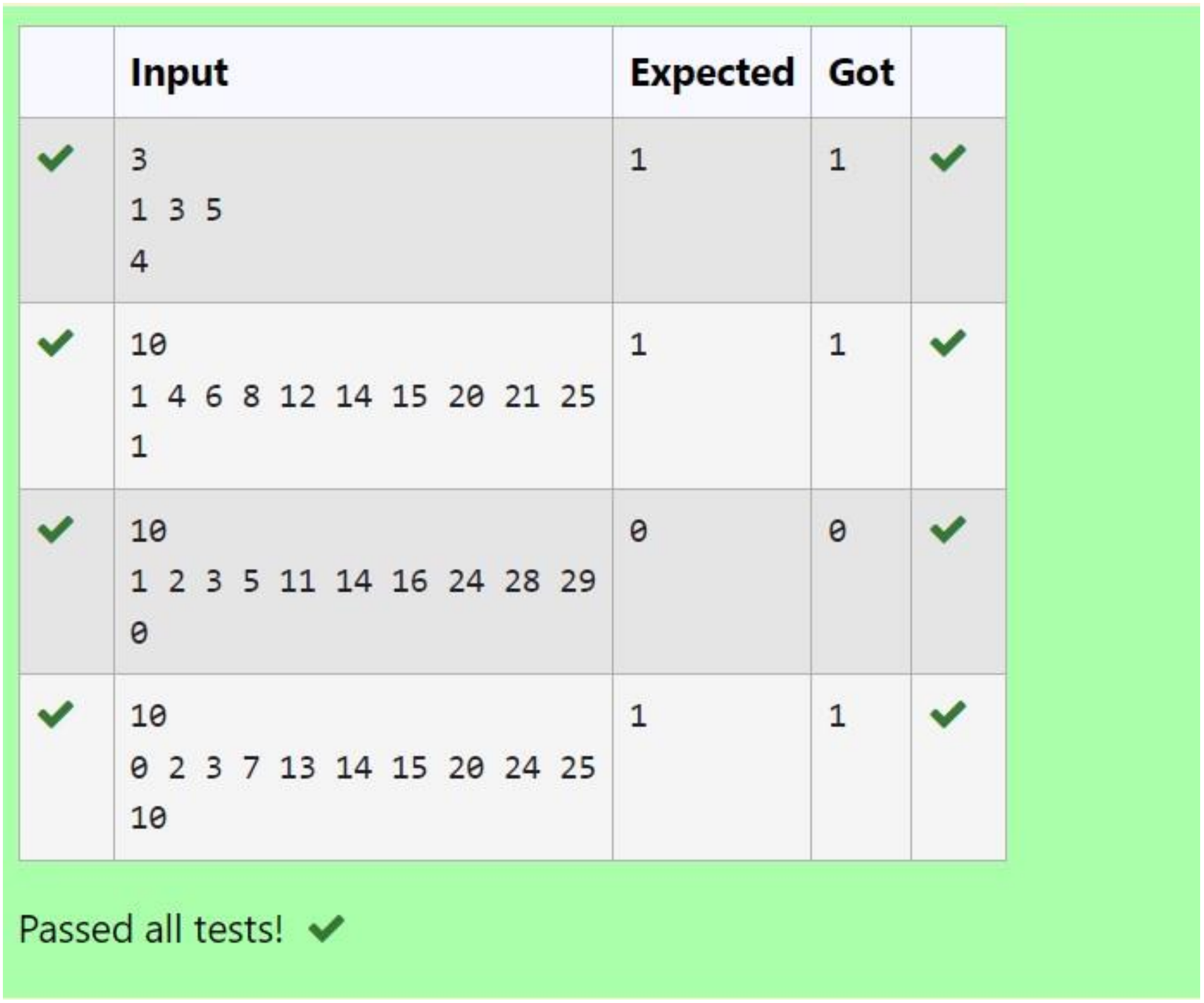
Adjust the pointers based on the value of the difference Step 5:
Print the result

PROGRAM:

```
#include <stdio.h>


int     checkpair(int     arr[],int
   n,int     k){     int     i=0,j=1;
   while(j<n){                    int
   diff=arr[j]arr[i]; if
      (diff==k &&
      i!=j){ return

         1;

      }

      else if(diff<k)
         { j++;

      }

      else{ i++;

      } if(i==j

         ){ j++;
```

```
        }
    }
    return 0;
}


int
    main()
    { int n,k;

    scanf("%d",
    &n); int arr[n]; for (int
    i=0;i<n;i++){
    scanf("%d",&arr[i]);

    }
    scanf("%d",&k);
    int result=checkpair(arr,n
    ,k); printf("%d\n",result);
}
```

<u>OUTPUT:</u>



	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! ✓

RESULT: **Thus the program executes successfully.**