RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23331
DESIGN AND ANALYSIS OF ALGORITHM

Laboratory Observation NoteBook

Name : SRI AKASH U G
Year/Branch/Section : II/CSE/F
Register No. : 230701336
Semester : III
Academic Year: 2024-25

01 - BASIC C PROGRAMMING

Ex. No. : 1.1 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Given two numbers, write a c program to swap two numbers.

For example:

INPUT	RESULT
10 20	20 10

ALGORITHM:

Step 1: Start

Step 2: Get two numbers from the user, a and b.

Step 3: Store the value of a in a temporary variable called temp.

Step 4: Change the value of a to be the value of b. Set the value of b to the value stored in temp.

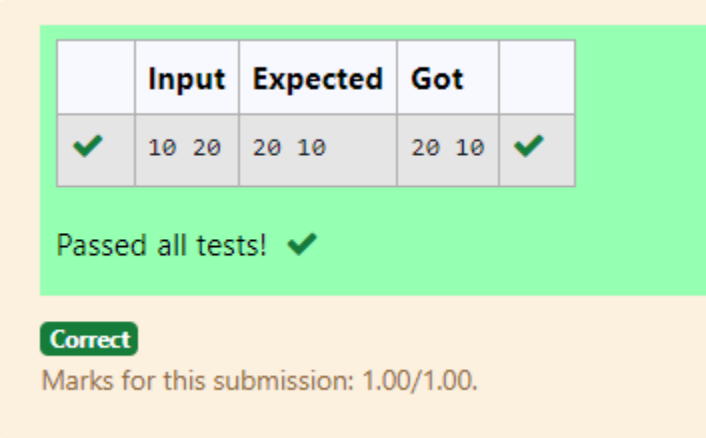
Step 5: Show the new values of a and b on the screen.

Step 6: End

PROGRAM:

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

OUTPUT:



	Input	Expected	Got	
✔	10 20	20 10	20 10	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.2 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Write a c program to check the eligibility of admission 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 CASE:

1. INPUT: 70 60 80

   OUTPUT: The candidate is eligible

2. INPUT: 50 60 40

   OUTPUT: The candidate is not eligible

ALGORITHM:

Step 1: Start

Step 2: Read the marks for three subjects (m, p, c) from the user.

Step 3: Calculate the total marks, tot = m + p + c.

Step 4: Check if (m >= 65 and p >= 55 and c >= 50) or tot >= 180.

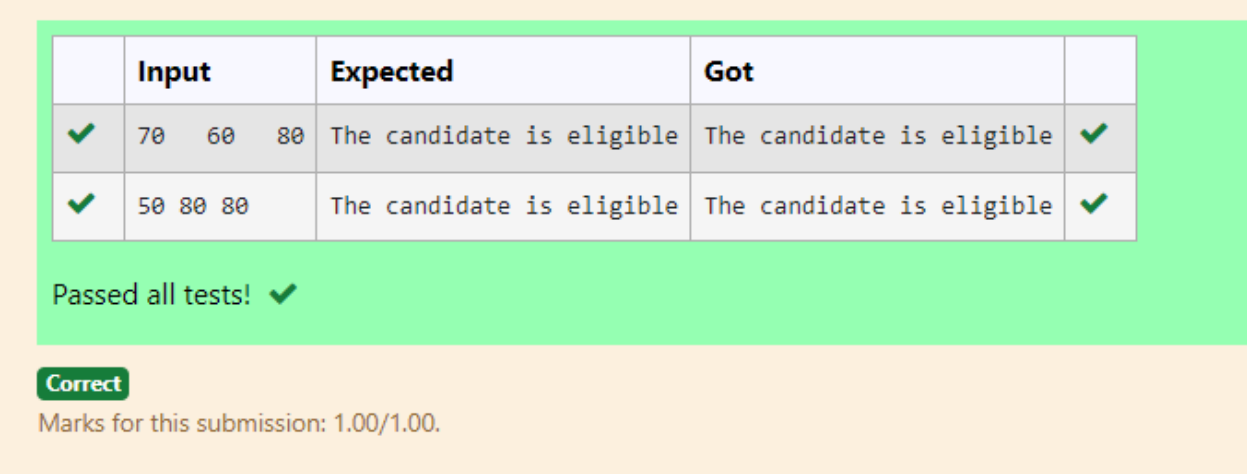
Step 5: If the condition is true, print "The candidate is eligible", otherwise print "The candidate is not eligible".

Step 6: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int m,p,c,ans=0;
    scanf("%d%d%d",&m,&p,&c);
    ans=m+p+c;;
    if((m>=65 && p>=55 && c>=50)||ans>=180)
    {
        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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.3 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Malini goes to BestSave supermarket to buy grocery. Supermarket provides 10% discount on bill amount B when the B is more than Rs.2000.

The program must find the final payable amount A.

SAMPLE TEST CASE:

1. INPUT: 1900

   OUTPUT: 1900

2. INPUT: 3000

   OUTPUT: 2700

ALGORITHM:

Step 1: Start

Step 2: Read the value of b from the user.

Step 3: If b > 2000, calculate 10% of b and store it in a (a = b * 0.1), otherwise remains uninitialized.

Step 4: Subtract a from b to calculate the final amount.

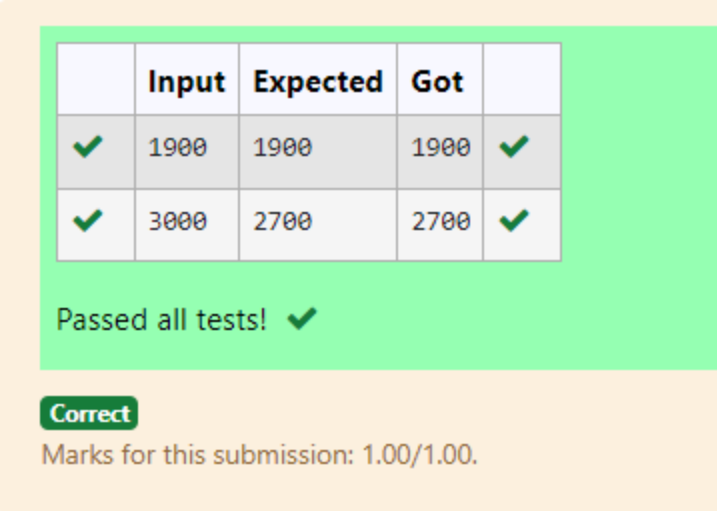
Step 5: Print the RESULT (b - a).

Step 6: End

PROGRAM:

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

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.4 **Date: 12.08.24**

AIM:

Baba is very kind to beggars and everyday baba donates half of the amount he has when ever the beggar requests him.The money M left in Baba's hand is passed as the input and the numbers of beggars B is input. The program must print the money baba had in the beginning of the day.

SAMPLE TEST CASE:

1. INPUT: 100

2

OUTPUT: 400

Explaination:

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of m from the user.

Step 3: Read the value of b from the user.

Step 4: Calculate t = 2 * m * b.

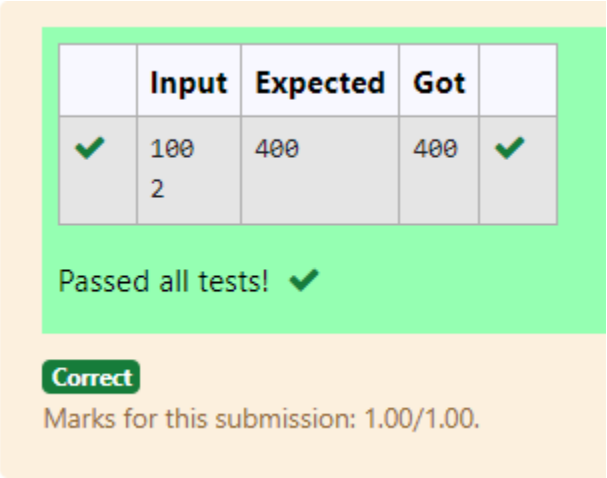
Step 5: Print the value of t.

Step 6: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int m,b,money=0;
    scanf("%d%d",&m,&b);
    money=m*b*2;
    printf("%d",money);
}
```


OUTPUT:



	Input	Expected	Got	
✔	100 2	400	400	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.5 **Date: 12.08.24**

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 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.

SAMPLE TEST CASE:

1. INPUT: 500

3

OUTPUT: 2100

Explanation:

On monday the employee receives Rs.500, on tuesday Rs.700, on wednesday Rs.900. So total=Rs.2100

ALGORITHM:

Step 1: Start

Step 2: Read the value of s from the user and initialize r = s.

Step 3: Read the value of n from the user.

Step 4: Loop from i = 0 to n-2:

a. Add 200 to r (r = r + 200).

b. Add the updated value of r to s (s = s + r).

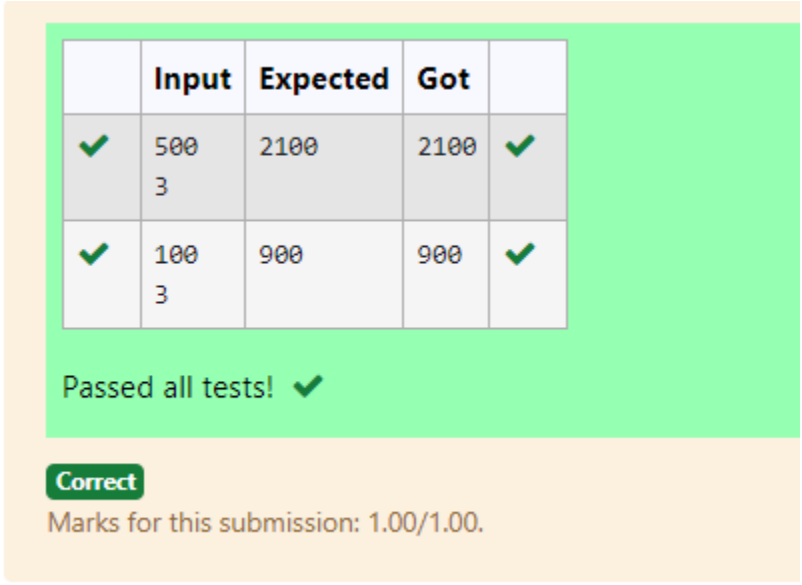
Step 5: Print the final value of s.

Step 6: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int i,n,pi=0;
    scanf("%d%d",&i,&n);
    while(n>0){
        pi=pi+i;
        i=i+200;
        n--;
    }
    printf("%d",pi);
}
```

OUTPUT:



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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.6 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

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.

SAMPLE TEST CASE:

1. INPUT: 2

40

7

OUTPUT: 35 28 21 14 7

ALGORITHM:

Step 1: Start

Step 2: Read the values of m, n, and x from the user.

Step 3: Loop through i from n to m (decreasing):

a. If i is divisible by x (i % x == 0), print i.

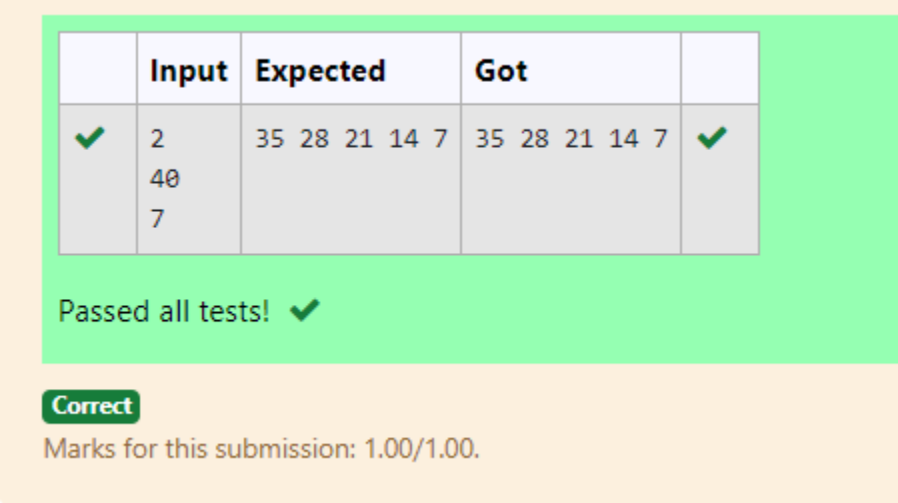
b. Otherwise, continue the loop.

Step 4: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int m,n,x;
    scanf("%d%d%d",&m,&n,&x);
    for(int 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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.7 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Write a c program to find the quotient and reminder of given number.

For example:

INPUT	RESULT
12 3	4 0

ALGORITHM:

Step 1: Start

Step 2: Read the values of n and m from the user.

Step 3: Calculate the quotient by performing integer division n / m and print the RESULT.

Step 4: Calculate the remainder by performing n % m and print the RESULT.

Step 5: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int a,b,quo=0,rem=0;
    scanf("%d%d",&a,&b);
    quo=a/b;
    rem=a%b;
    printf("%d\n%d",quo,rem);
}
```

OUTPUT:

	Input	Expected	Got	
✔	12 3	4 0	4 0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.8 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Write a C program to find the biggest among the given 3 integers.

For example:

INPUT	RESULT
10 20 30	30

ALGORITHM:

Step 1: Start

Step 2: Read the values of n1, n2, and n3 from the user.

Step 3: Check if n1 is greater than both n2 and n3. If true, print n1.

Step 4: Otherwise, check if n2 is greater than both n1 and n3. If true, print n2.

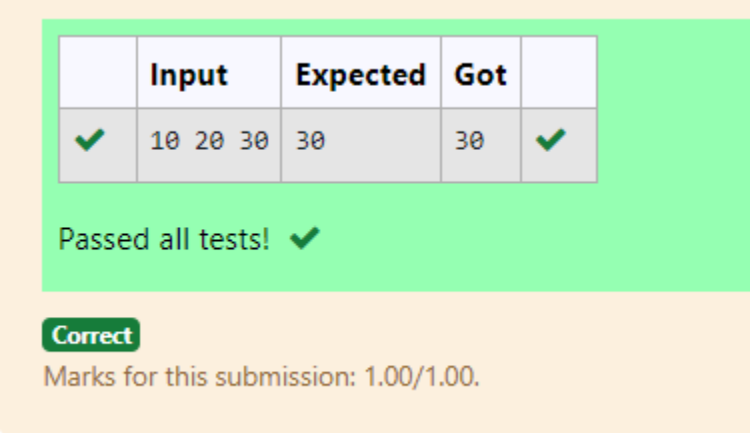
Step 5: If neither of the above conditions is true, print n3.

Step 6: End

PROGRAM:

```c
#include<stdio.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{
        printf("%d",c);
    }
}
```

OUTPUT:



	Input	Expected	Got	
✔	10 20 30	30	30	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.9 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Write a C program to find whether the given integer is odd or even.

For example:

INPUT	RESULT
12	Even
11	Odd

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Check if n is even by evaluating the condition n % 2 == 0.

Step 4: If the condition is true, print "Even".

Step 5: If the condition is false, print "Odd".

Step 6: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int a;
    scanf("%d",&a);
    if(a%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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.10 **Date: 12.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

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

For example:

INPUT	RESULT
5	120

ALGORITHM:

Step 1: Start
Step 2: Read the value of n from the user.
Step 3: Initialize a variable f to 1.
Step 4: Loop through i from 2 to n:
    a. Multiply f by i (f = f * i).
Step 5: Print the value of f, which is the factorial of n.
Step 6: End

PROGRAM:

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

OUTPUT:

	Input	Expected	Got	
✔	5	120	120	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.11 **Date: 12.08.24**

AIM:

Write a C program to find the sum of first N natural numbers.

For example:

INPUT	RESULT
3	6

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Initialize a variable s to 0.

Step 4: Loop through i from 1 to n:

a. Add i to s (s = s + i).

Step 5: Print the value of s, which is the sum of the first n natural numbers.

Step 6: End

PROGRAM:

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

OUTPUT:

	Input	Expected	Got	
✔	3	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.12 **Date: 12.08.24**

AIM:

Write a C program to find the Nth term in fibonacci series.

For example:

INPUT	RESULT
0	0
1	1
4	3

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Initialize f0 = 0, f1 = 1, and f2.

Step 4: If n is 0, print f0; if n is 1, print f1.

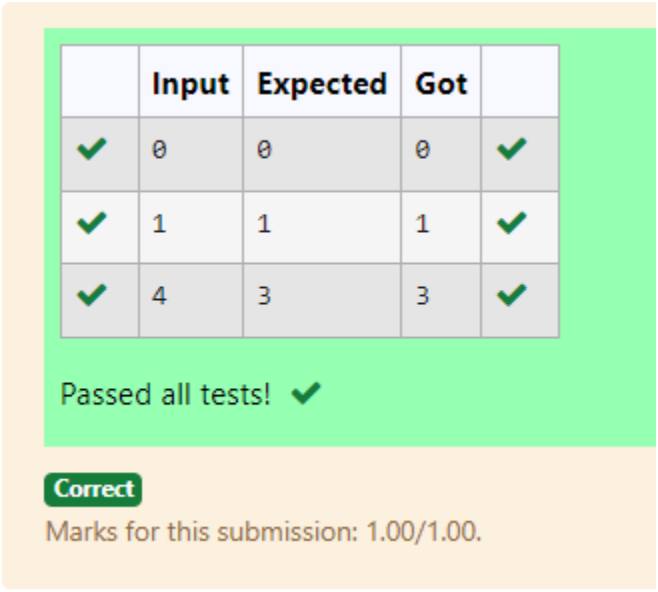
Step 5: For i from 1 to n - 1, calculate f2 = f1 + f0, then update f0 to f1 and f1 to f2.

Step 6: Print the value of f2. End.

PROGRAM:

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

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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.13 **Date: 12.08.24**

AIM:

Write a C program to find the power of integers.

For example:

INPUT	RESULT
2 5	32

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and store it in t.

Step 3: Read the value of m from the user.

Step 4: For i from 1 to m - 1, multiply n by t.

Step 5: Print the final value of n.

Step 6: End

PROGRAM:

```c
#include<stdio.h>
#include<math.h>
int main()
{
    int a,b,ans=0;
    scanf("%d%d",&a,&b);
    ans=pow(a,b);
    printf("%d",ans);
}
```

OUTPUT:

	Input	Expected	Got	
✔	2 5	32	32	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.14 **Date: 12.08.24**

AIM:

Write a C program to find whether the given integer is prime or not.

For example:

INPUT	RESULT
7	Prime
9	No Prime

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Initialize a variable flag to 0.

Step 4: For i from 2 to n - 1, check if n % i == 0. If true, set flag to 1 and break the loop.

Step 5: If flag is 0, print "Prime"; otherwise, print "No Prime".

Step 6: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int n,flag=1;
    scanf("%d",&n);
    for(int i=2;i<n/2;i++)
    {
        if(n%i==0){
            flag=0;
            break;
        }
        else{
            flag=1;
        }
    }
    if(flag==1)
    {
        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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 1.15 **Date: 12.08.24**

AIM:

Write a C program to find the reverse of the given integer.

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Initialize a variable d to 0.

Step 4: While n is not 0, do the following:

a. Set r to n % 10.

b. Update d as d = (d * 10) + r.

c. Update n as n = n / 10.

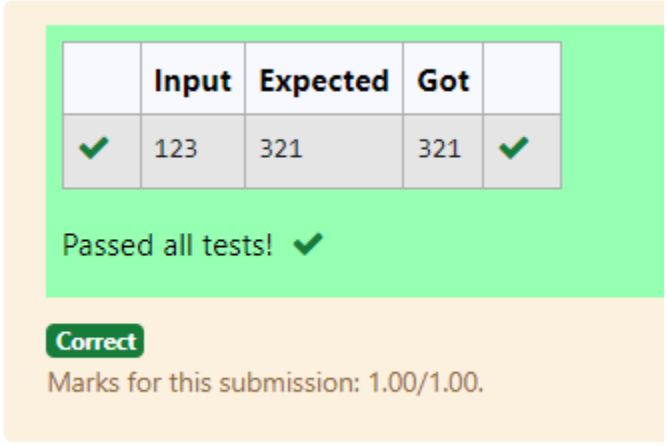
Step 5: Print the value of d, which is the reversed number.

Step 6: End

PROGRAM:

```c
#include<stdio.h>
int main()
{
    int n,a=0;
    scanf("%d",&n);
    while(n>0)
    {
        int rem=n%10;
        a=(a*10)+rem;
        n=n/10;
    }
    printf("%d",a);

}
```

OUTPUT:



	Input	Expected	Got	
✔	123	321	321	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

02 - Finding Time Complexity of Algorithms

Ex. No. : 2.1 **Date: 20.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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;
     }
}
```

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Call the function func(n), initializing count, i to 1, and s to 1. Increment count (1st increment).

Step 4: While s <= n, increment count (2nd increment), then increment i and update s by adding i. Increment count again (3rd increment).

Step 5: After exiting the loop, increment count (4th increment).

Step 6: Print the value of count.

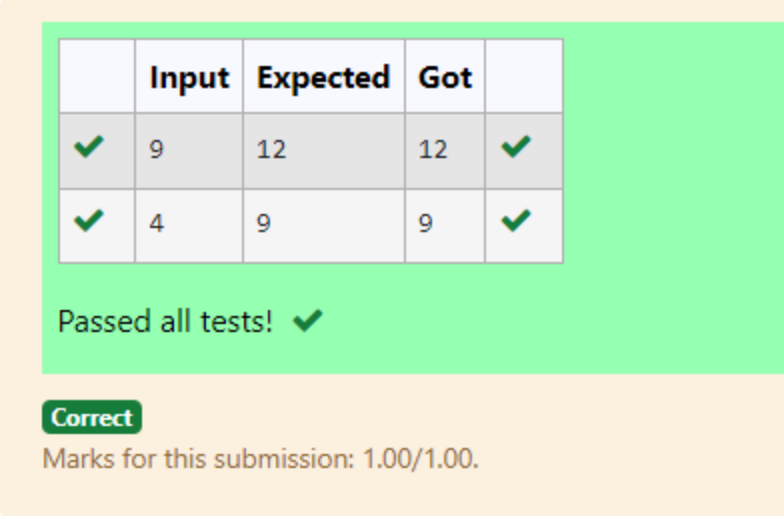
Step 7: End

PROGRAM:

```
#include<stdio.h>

void function(int n)

{

   int count=0;

   int i=1;

   count++;

   int s=1;

   count++;

   while(s<=n)

   {

      i++;
```

```
        count++;
        s+=i;
        count++;
        count++;
    }
    count++;
    printf("%d",count);
}
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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 2.2 **Date: 20.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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;
       }}}}
```

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Call the function func(n).

Step 4: In func, if n == 1, increment count (1st increment).

Step 5: If n > 1, increment count (2nd increment) and loop i from 1 to n, increment count (3rd increment) for each iteration, and loop j from 1 to n, incrementing count (4th increment) three times, then break. Increment count (5th increment) after the inner loop, and once more after the outer loop (6th increment).

Step 6: Print the value of count.

Step 7: End

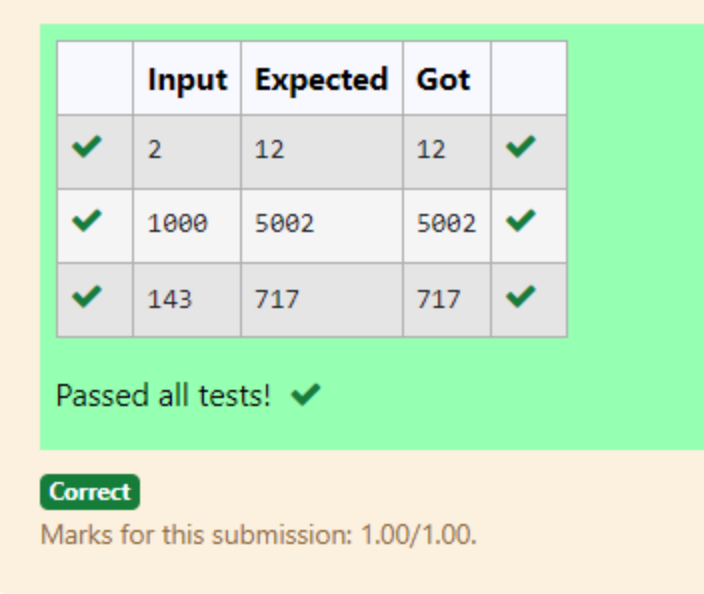
PROGRAM:

```
#include<stdio.h>

void func(int n)
```


```
{
    int count=0;
    if(n==1)
    {
        count++;
    }
    else
    {
        count++;
        for(int i=1;i<=n;i++)
        {
            count++;
            for(int j=1;j<=n;j++)
            {
                count++;
                count++;
                count++;
                break;
            }
            count++;
```

```
        }
        count++;
    }
    printf("%d",count);
}

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: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 2.3 **Date: 20.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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);
    }
     }
}
```

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Call the function Factor(n).

Step 4: In Factor, loop i from 1 to num, increment count (1st increment).

Step 5: For each i, check if num % i == 0. If true, increment count (2nd increment). Increment count again (3rd increment) for the end of the loop.

Step 6: After the loop, increment count (4th increment).

Step 7: Print the value of count.

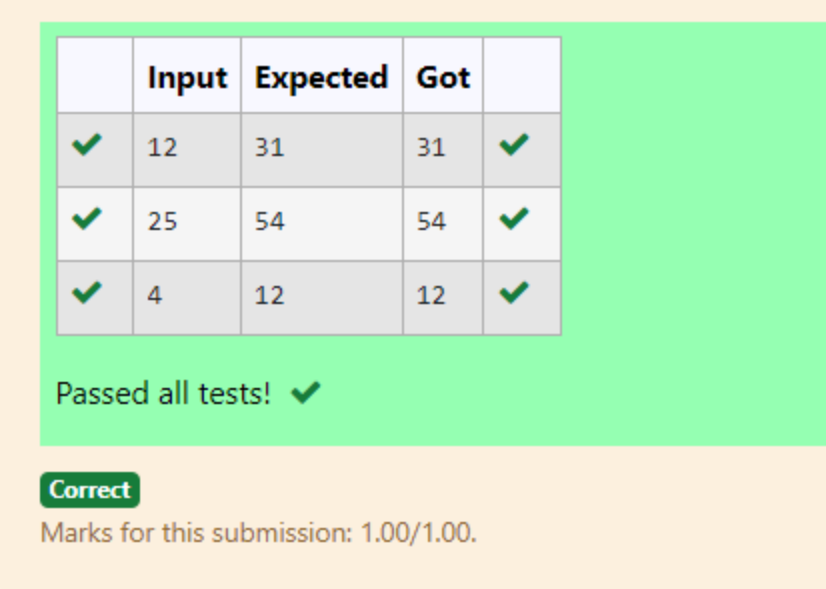
Step 8: End

PROGRAM:

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

```
void Factor(int num)
{
    int i,count=0;
    for(i=1;i<=num;i++)
    {
        count++;
        if(num%i==0)
        {
            count++;
        }
        count++;
    }
    count++;
    printf("%d",count);
}
int main()
{
    int num;
    scanf("%d",&num);
    Factor(num);
}
```

OUTPUT:



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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 2.4 **Date: 20.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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++;

}
```

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Call the function function(n).

Step 4: In function, initialize c to 0 and increment count (1st increment).

Step 5: Loop i from n/2 to n, incrementing count (2nd increment), and for each i, loop j from 1 to n, doubling j each time, incrementing count (3rd increment).

Step 6: Inside the j loop, loop k from 1 to n, doubling k each time, incrementing count (4th increment), increment c, and increment count (5th increment). Increment count again after the k loop (6th increment) and after the j loop (7th increment).

Step 7: Increment count after the i loop (8th increment).

Step 8: Print the value of count.

Step 9: End

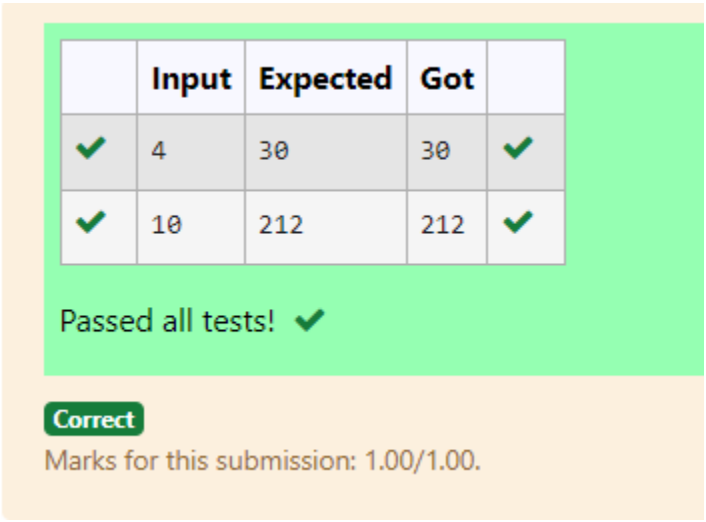
PROGRAM:

```
#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)
            {
                c++;
                count++;
                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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 2.5 **Date: 20.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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);

}
```

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user.

Step 3: Call the function reverse(n).

Step 4: In reverse, initialize rev to 0 and increment count (1st increment).

Step 5: While n is not 0, increment count (2nd increment), calculate remainder as n % 10, and increment count (3rd increment). Update rev by multiplying it by 10 and adding remainder, then increment count (4th increment). Divide n by 10 and increment count (5th increment).

Step 6: After exiting the loop, increment count (6th increment) and again for the commented print statement (7th increment).

Step 7: Print the value of count.

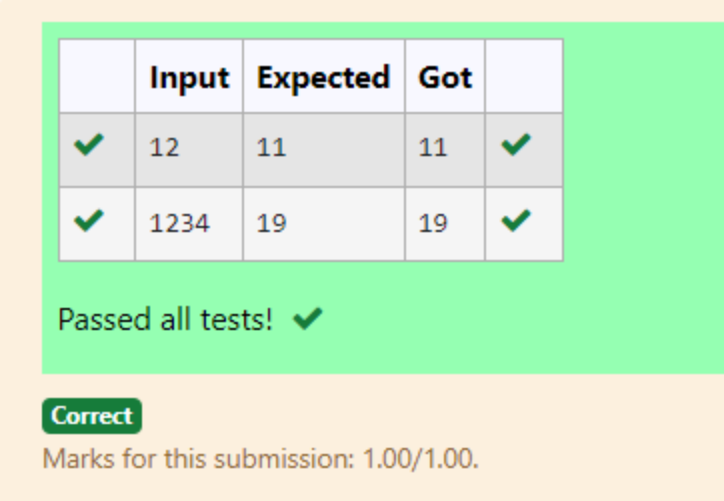
Step 8: End

PROGRAM:

```
#include<stdio.h>

void reverse(int n)

{

   int count=0;

   int rev=0;

   count++;

   int remainder;

   count++;

   while(n!=0)
```

```
    {
        count++;
        remainder=n%10;
        count++;
        rev=rev*10+remainder;
        count++;
        n/=10;
        count++;
    }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.

RESULT:

Hence the above program has been executed successfully.

03 - Greedy Algorithms

Ex. No. : 3.1 **Date: 26.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

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.

ALGORITHM:

Step 1: Start

Step 2: Initialize an array currency with denominations and read the value of n from the user.

Step 3: Initialize a variable count to 0 and a variable j to 0.

Step 4: Use a while loop to find the first denomination in currency that is less than or equal to n.

Step 5: While n is not 0, check if the current currency[j] is less than or equal to n. If true, update count by adding the integer division of n by currency[j], then update n using the remainder of that division. Increment j.

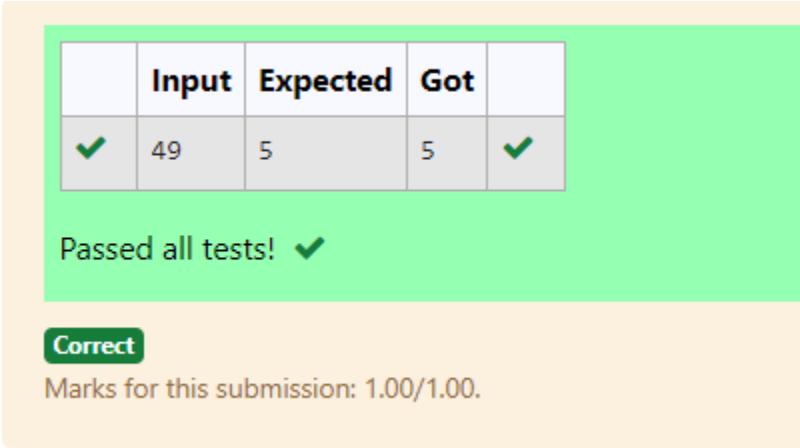
Step 6: Print the value of count, which represents the total number of currency notes/coins used.

Step 7: End

PROGRAM:

```
#include<stdio.h>
int main()
{
    int currency[]={1000,500,100,50,20,10,5,2,1};
    int n,count=0;
    scanf("%d",&n);
    int j = 0;
        while(currency[j]>n){
            j++;
        }
        while(n!=0){
            if(currency[j]<n){
                count+=n/currency[j];
                n=n%currency[j];
            }
            j++;
        }
    printf("%d",count);
}
```


OUTPUT:



	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 3.2 **Date: 26.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

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.

ALGORITHM:

Step 1: Start

Step 2: Read the value of c (the number of groups) from the user. Initialize an array g of size c and read c values into it.

Step 3: Read the value of n (the number of elements) from the user. Initialize an array s of size n and read n values into it.

Step 4: Initialize a variable count to 0.

Step 5: Use a nested loop to compare each element in g with elements in s. If any s[j] is greater than or equal to g[i], increment count and break the inner loop.

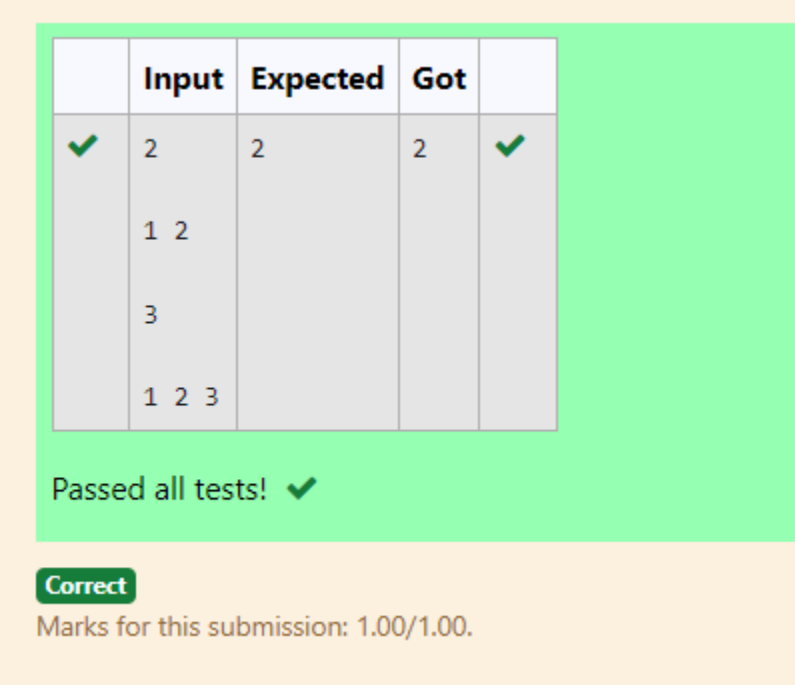
Step 6: Print the value of count, representing the number of successful comparisons.

Step 7: End

PROGRAM:

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


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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 3.3 **Date: 26.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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 $3^i * c$ 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 burgers 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-separated 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
```

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array cal of size n. Read n values into the array.

Step 3: Use a nested loop to sort the array cal in descending order using the bubble sort algorithm.

Step 4: Initialize a variable s to 0.

Step 5: Loop through the sorted array and calculate the value of s by summing pow(n, i) * cal[i] for each index i.

Step 6: Print the value of s.

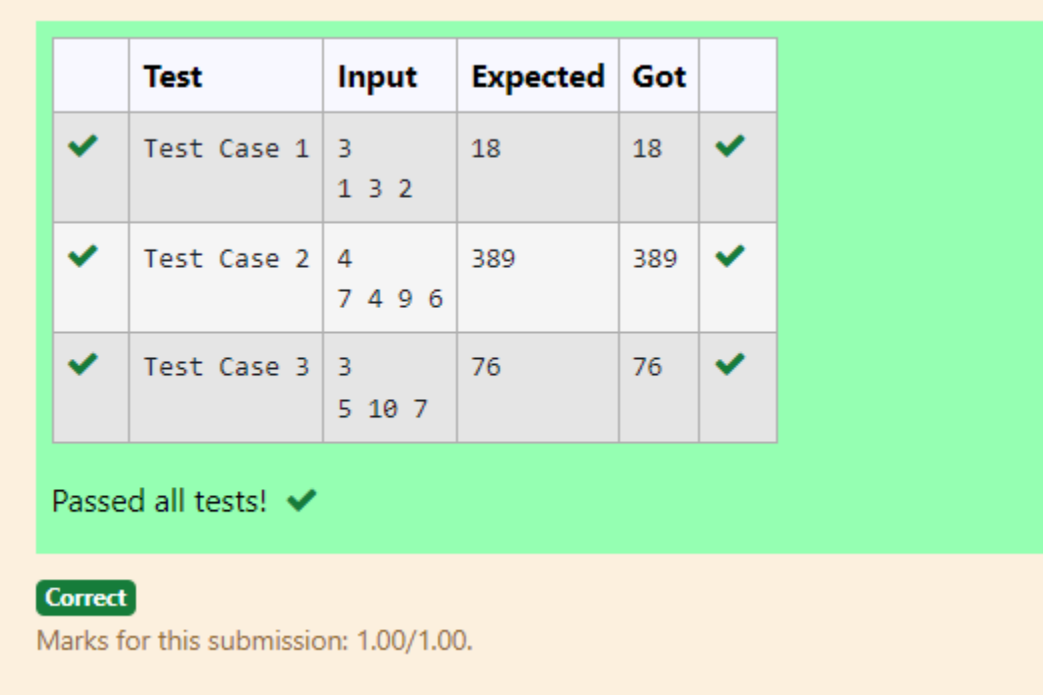
Step 7: End

PROGRAM:

```
#include<stdio.h>

#include<math.h>

int main()

{

    int n,km=0;

    scanf("%d",&n);

    int cal[n];

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

    {

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

```
    }
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-i-1;j++)
        {
            if(cal[j]<cal[j+1])
            {
                int temp=cal[j];
                cal[j]=cal[j+1];
                cal[j+1]=temp;
            }
        }
    }
    for(int i=0;i<n;i++)
    {
        km+=pow(n,i)*cal[i];
    }
    printf("%d",km);
}
```


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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 3.4 **Date: 26.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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).

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

ALGORITHM:

Step 1: Start

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

Step 3: Use the qsort function to sort the array arr in ascending order by calling the compare function.

Step 4: Initialize a variable s to 0.

Step 5: Loop through the sorted array and calculate the value of s by summing arr[i] * i for each index i.

Step 6: Print the value of s.

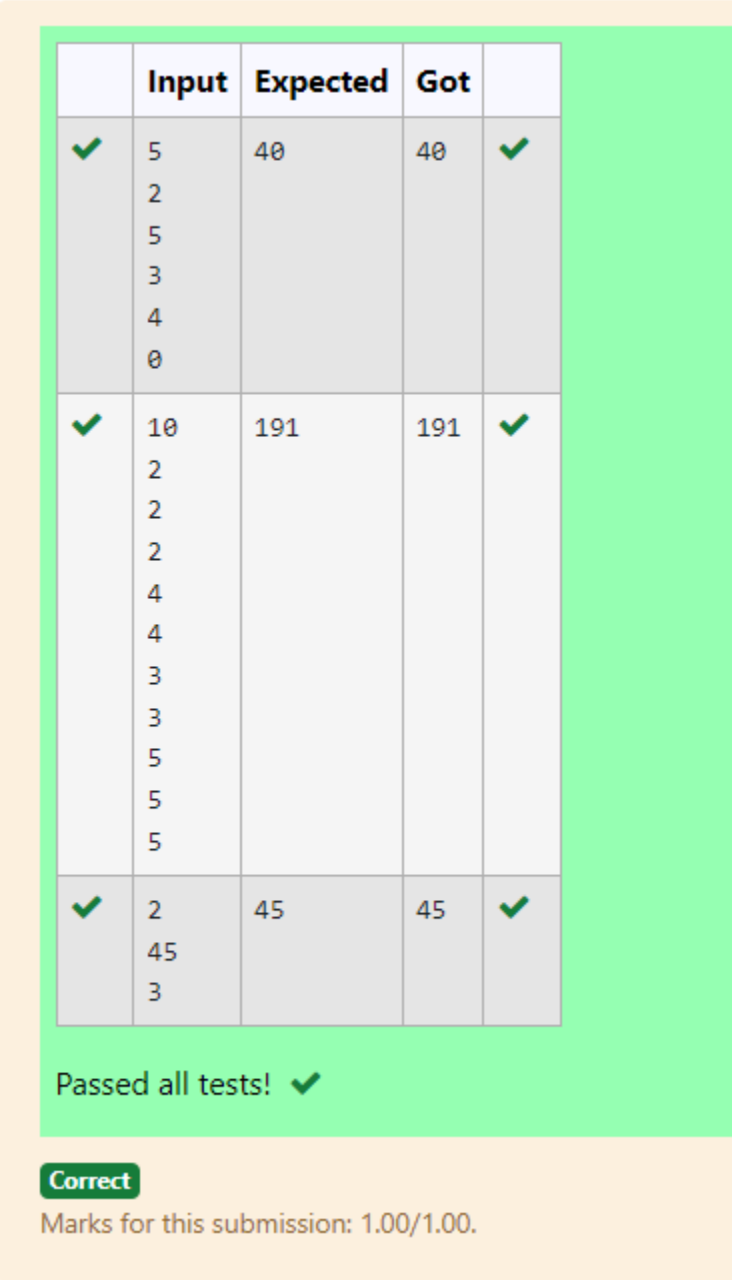
Step 7: End

PROGRAM:

```
#include<stdio.h>

int main()

{

    int n,sum=0;

    scanf("%d",&n);

    int arr[n];
```

```
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }for(int i=1;i<n;i++){
        int j=i;
        int temp=arr[j];
        while(j>0 && arr[j-1]>temp){
            arr[j]=arr[j-1];
            j--;
        }
        arr[j]=temp;}
    for(int i=0;i<n;i++){
        sum+=arr[i]*i;
    }
    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	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 3.5 **Date: 26.08.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

For example:

Input	RESULT
3 1 2 3 4 5 6	28

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize two arrays, arr1 and arr2, each of size n. Read n values into both arrays.

Step 3: Use the qsort function to sort arr1 in ascending order using compare1 and arr2 in descending order using compare2.

Step 4: Initialize a variable s to 0.

Step 5: Loop through both sorted arrays and calculate the value of s by summing arr1[i] * arr2[i] for each index i.

Step 6: Print the value of s.

Step 7: End

PROGRAM:

```
#include<stdio.h>
void sort(int a[],int x)
{
   for(int i=1;i<x;i++)
   {
      int j=i;
      int temp=a[j];
      while(j>0 && a[j-1]>temp)
      {
         a[j]=a[j-1];
         j--;
      }
      a[j]=temp;
   }
}
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 i=0;i<n;i++)
    {
        scanf("%d",&b[i]);
    }
    sort(a,n);
    sort(b,n);
    for(int i=0;i<n;i++)
    {
        sum+=a[i]*b[n-i-1];
    }
    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	✔
✔	5 20 10 30 10 40 8 9 4 3 10	590	590	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

04 - Divide and Conquer

Ex. No. : 4.1 **Date: 03.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

ALGORITHM:

Step 1: Start

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

Step 3: Check if the first element of arr is 0. If true, print n and exit the program.

Step 4: Call the divide function with arr, 0, and n-1 to find the index of the first occurrence of 0.

Step 5: If the index is not 0, print the value of n - index, which represents the count of 0s in the array. Otherwise, print the index.

Step 6: End

PROGRAM:

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

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	✔
✔	8 0 0 0 0 0 0 0 0 0	8	8	✔
✔	17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT :

Hence the above program has been executed successfully.

Ex. No. : 4.2 **Date: 03.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ 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
```

For example:

Input	RESULT
3 3 2 3	3
7 2 2 1 1 1 2 2	2

ALGORITHM:

Step 1: Start

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

Step 3: Use qsort to sort the array arr in ascending order.

Step 4: Loop through the array to find the first and last indices of each element using the first and last functions. Calculate the count of occurrences (major).

Step 5: If any element's count is greater than or equal to n/2, return that element.

Step 6: Print the element that appears more than n/2 times or print 0 if none is found.

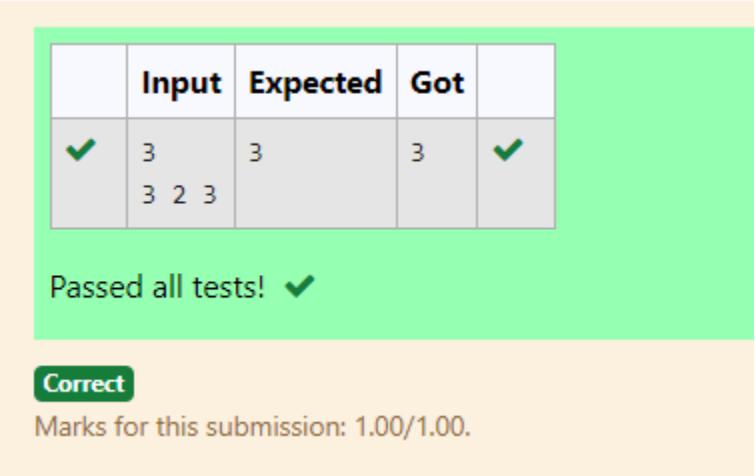
Step 7: End

PROGRAM:

```
#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)
    {
        Count(a,left,mid,key);
        Count(a,mid+1,right,key);
    }
    else
    {
        c++;
    }
    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! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT :

Hence the above program has been executed successfully.

Ex. No. : 4.3 **Date: 03.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

Problem Statement:
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.
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

ALGORITHM:

Step 1: Start

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

Step 3: Read the integer x from the user, which will be used to find the floor value.

Step 4: Call the search function with arr, x, 0, and n-1 to find the largest element in arr that is less than or equal to x.

Step 5: Print the floor value returned by the search function.

Step 6: End

PROGRAM:

```
#include<stdio.h>
int search(int[],int,int,int);
int search(int arr[],int x,int left,int right)
{
   int mid=left+(right-left)/2;
    if(arr[mid]<=x)
      {
         int max = arr[mid];
         for(int i=0;i<mid;i++){
            if(arr[i]>=max)
               max=arr[i];
         }
```

```
        return max;
      }
    else if(arr[mid]>x)
    {
      return search(arr,x,left,mid);
      }
   else
      return search(arr,x,mid+1,right);
}

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

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 13 15 10	9	9	✔

Passed all tests! ✔

RESULT :

Hence the above program has been executed successfully.

Ex. No. : 4.4 **Date: 03.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

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")

ALGORITHM:

Step 1: Start

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

Step 3: Read the integer x from the user, which represents the target sum.

Step 4: Call the twosum function with arr, 0, n-1, and x to find two numbers in the array that add up to x.

Step 5: If a pair is found, print the two numbers; otherwise, print "No" to indicate that no such pair exists.

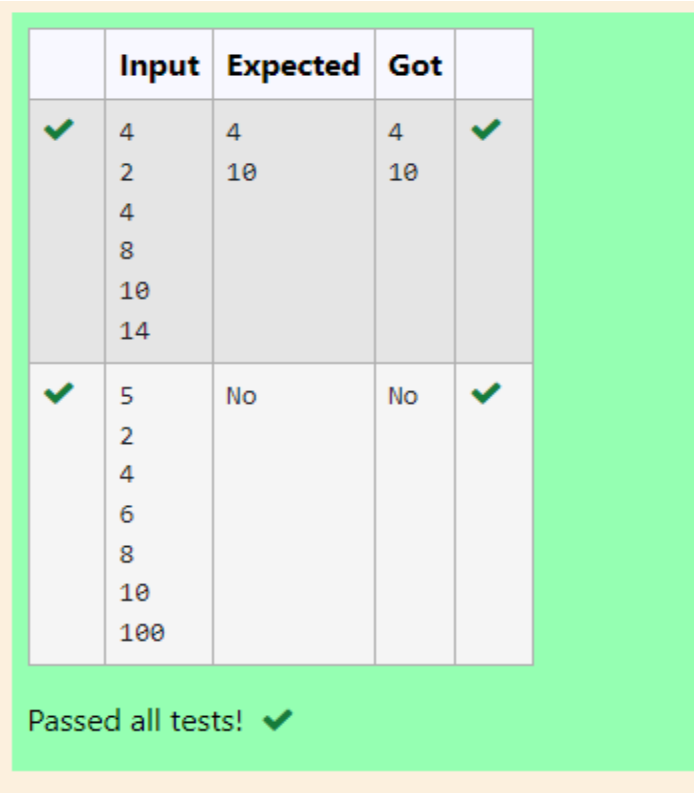
Step 6: End

PROGRAM:

```
#include<stdio.h>

void twosum(int arr[],int left,int right,int x){

    if (left >= right){

        printf("No");

        return;}

    int sum=arr[left]+arr[right];

    if (sum==x){

        printf("%d\n",arr[left]);
```

```
        printf("%d\n",arr[right]);
    }
    else if(sum<x){
        twosum(arr,left+1,right,x);
    }
    else{
        twosum(arr,left,right-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);
    twosum(arr,0,n-1,x);
    return 0;
}
```


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 :

Hence the above program has been executed successfully..

Ex. No. : 4.5 **Date: 03.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and dynamically allocate an array arr of size n. Read n values into the array.

Step 3: Call the q_sort function with arr, 0, and n-1 to sort the array using the Quick Sort algorithm.

Step 4: In the q_sort function, select a pivot and partition the array into two halves. Recursively apply the same sorting process to both halves.

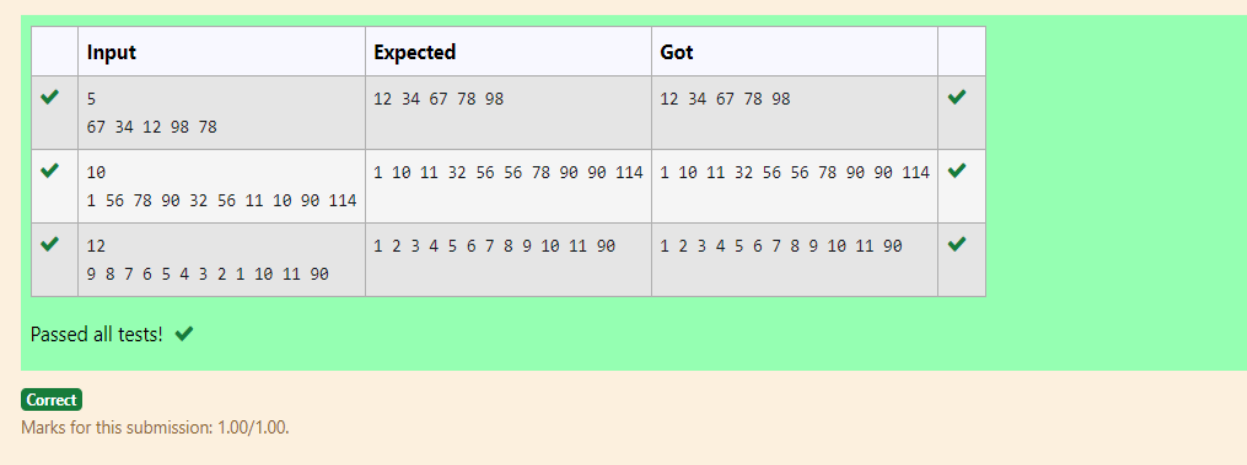
Step 5: Once sorted, iterate through the array and print the sorted values.

Step 6: End

PROGRAM:

```
#include<stdio.h>
void quicksort(int arr[],int left,int right){
    if(left<right){
        int j=right;
        int i=left;
        int pivot=left;
        while(i<j){
            while(arr[i]<=arr[pivot]){
                i++;}
            while(arr[j]>arr[pivot]){
                j--; }
            if(i<j){
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
```

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.

RESULT :

Hence the above program has been executed successfully..

05 - Dynamic Programming

Ex. No. : 5.1 **Date: 10.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

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

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<stdio.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;
}
```

OUTPUT:

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	243828195967216 29	243828195967216 29	✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 5.2 **Date: 10.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

Example:
Input
3
1 2 4
2 3 4
8 7 1
Output:
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

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[i-1][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	✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 5.3 **Date: 10.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

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

Example:

s1: ggtabe

s2: tgatasb

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

The length is 4

Solving it using Dynamic Programming

For example:

Input	Result
aab azb	2

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<string.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	✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 5.4 **Date: 10.09.24**

AIM:

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

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	✔

RESULT :

Hence the above program has been executed successfully..

06 - Competitive Programming

Ex. No. : 6.1 **Date: 17.09.24**

AIM:

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array arr of size n. Read n integer values into the array.

Step 3: Iterate through each element in the array. For each element, set temp to the current element and initialize j to the current index.

Step 4: Inside a while loop, check if the current element temp is equal to the next element in the array. If they are equal, set flag to 1 and break out of the loop.

Step 5: If flag is 0 after the inner loop, continue to the next iteration. If flag is 1, print the duplicate value temp and break the outer loop.

Step 6: End

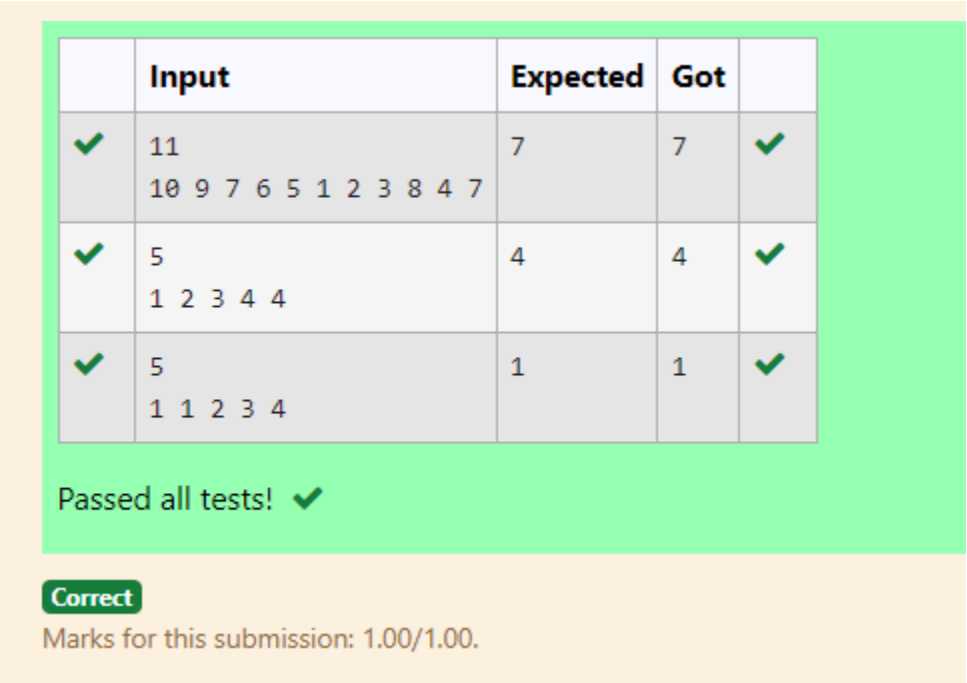
PROGRAM:

```
#include<stdio.h>

int main()

{

    int n,a,flag=0;

    scanf("%d",&n);

    int arr[n];
```


```
for(int i=0;i<n;i++){
    scanf("%d",&a);
    arr[i]=a;
}
for(int i=0;i<n;i++)
{
    int j=i;
    int temp=arr[j];
    while(j<n)
    {
        if(temp==arr[j+1]){
            flag=1;
            break;
        }
        else
            j++;
    }
    if(flag==0)
        continue;
    else{
```

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

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.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 6.2 **Date: 17.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

AIM:

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

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].

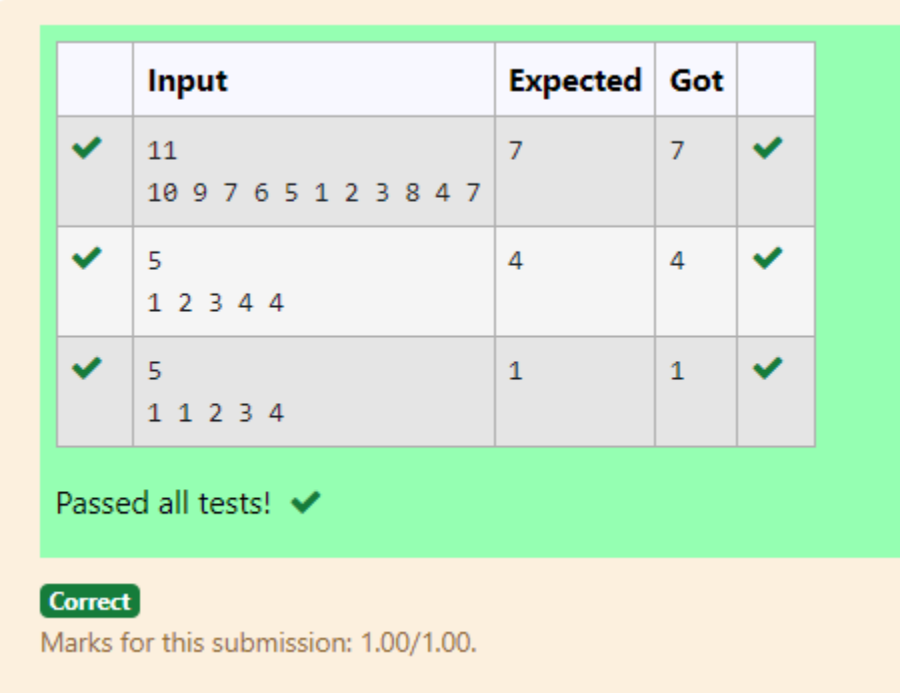
Step 5: After the loop, print the value of t.

Step 6: End

PROGRAM:

```
#include<stdio.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! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 6.3 **Date: 17.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

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

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]==element) {
                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",&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	✔

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 6.4 **Date: 17.09.24**

Register No.: 230701336 **Name: SRI AKASH U G**

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.

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

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	✔

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 6.5 **Date: 17.09.24**

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.

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.

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	✔

RESULT:

Hence the above program has been executed successfully.

Ex. No. : 6.6 **Date: 17.09.24**

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.

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.

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);
}
```

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	✔

RESULT:

Hence the above program has been executed successfully.