NAME: SREYA G **REGISTER NO.:**
230701334

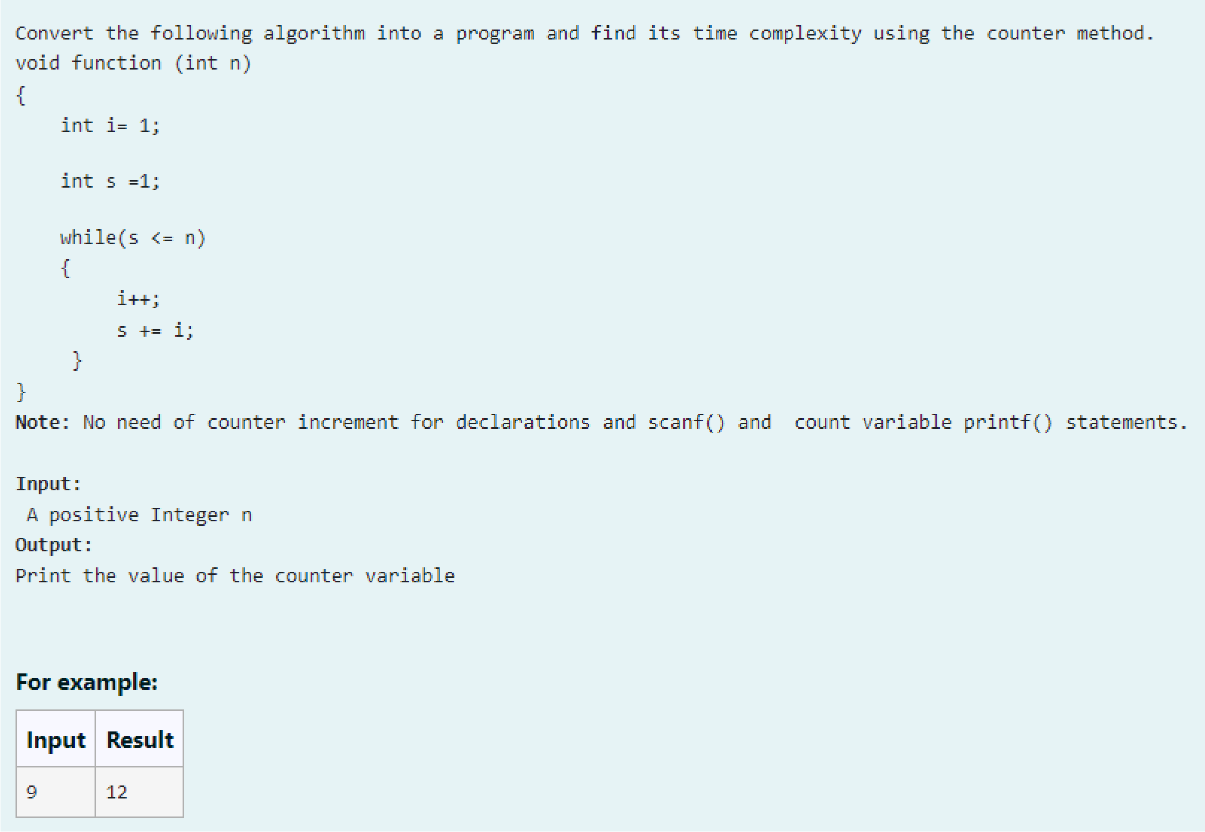
CLASS: CSE F **DATE:**
10/08/2024

EX – 2:

FINDING TIME COMPLEXITY OF ALGORITHMS:

PROBLEM 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;
     }
}
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:

1. Read input n.

2. Initialize counter = 0, i = 1, and s = 1.

3. Increment counter for each initialization.

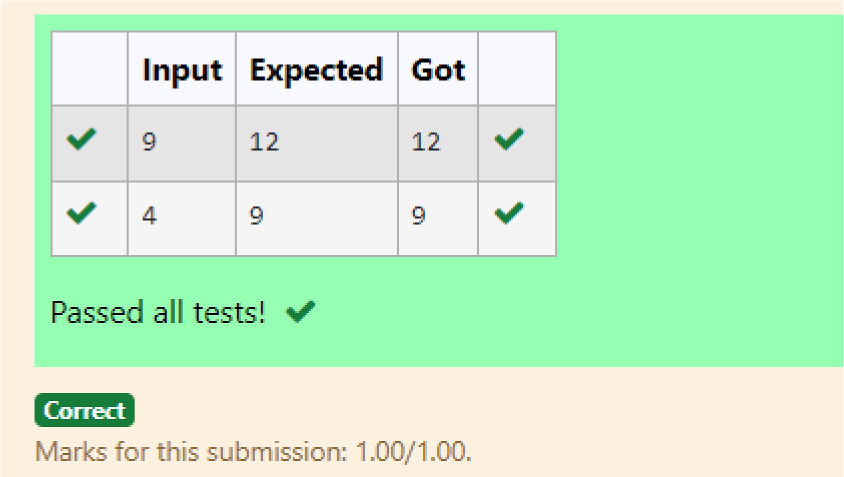
4. While s <= n:

```
   - Increment counter.
   - Increment i and update s = s + i.
   - Increment counter for each operation.
5. Increment counter for the final loop check.
6. Print counter.
7. End.
```

CODE:

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

OUTPUT:



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

Passed all tests! ✓

Correct

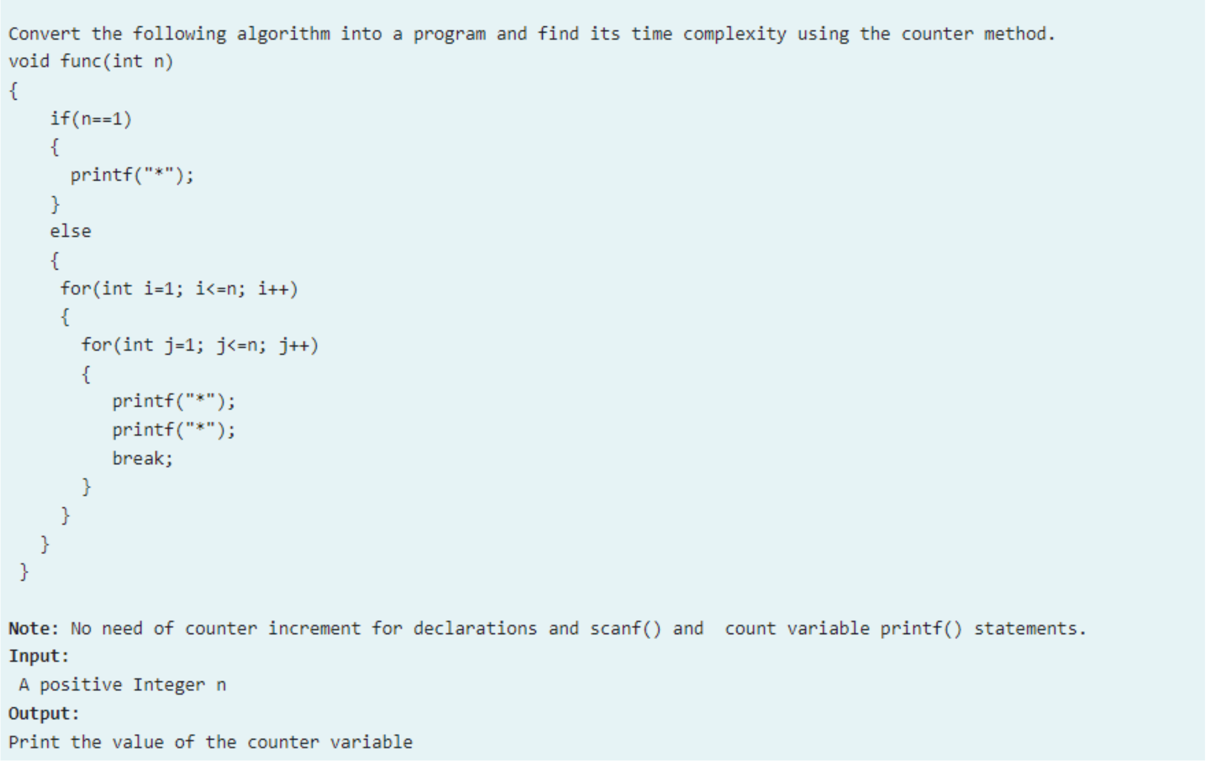
Marks for this submission: 1.00/1.00.

RESULT:

Thus the above code is executed successfully and gives the expected output.

PROBLEM 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;
       }
     }
   }
 }
```

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:

1. Input n.

2. Initialize count = 0.

3. If n == 1:

   - Increment count.

   - Print "*".

4. Else:

   - Increment count.

   - For each i from 1 to n:

     - Increment count.

     - For each j from 1 to n:

       - Increment count for operations and break.

     - Increment count after the inner loop.

   - Increment count after the outer loop.

5. Print count.

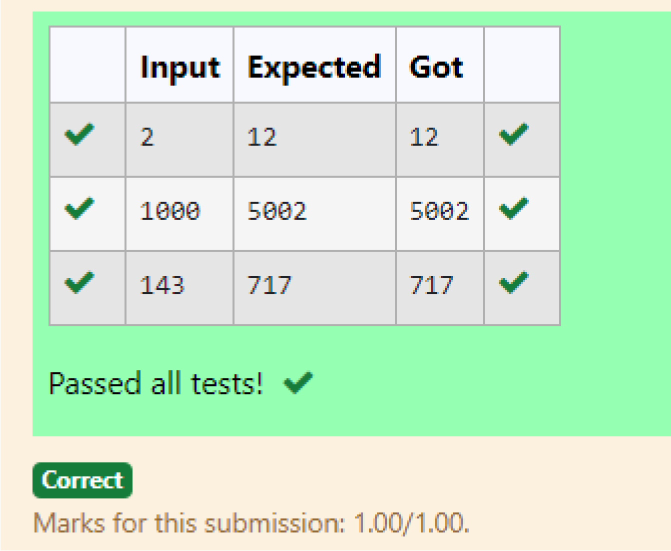
6. End.

CODE:

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

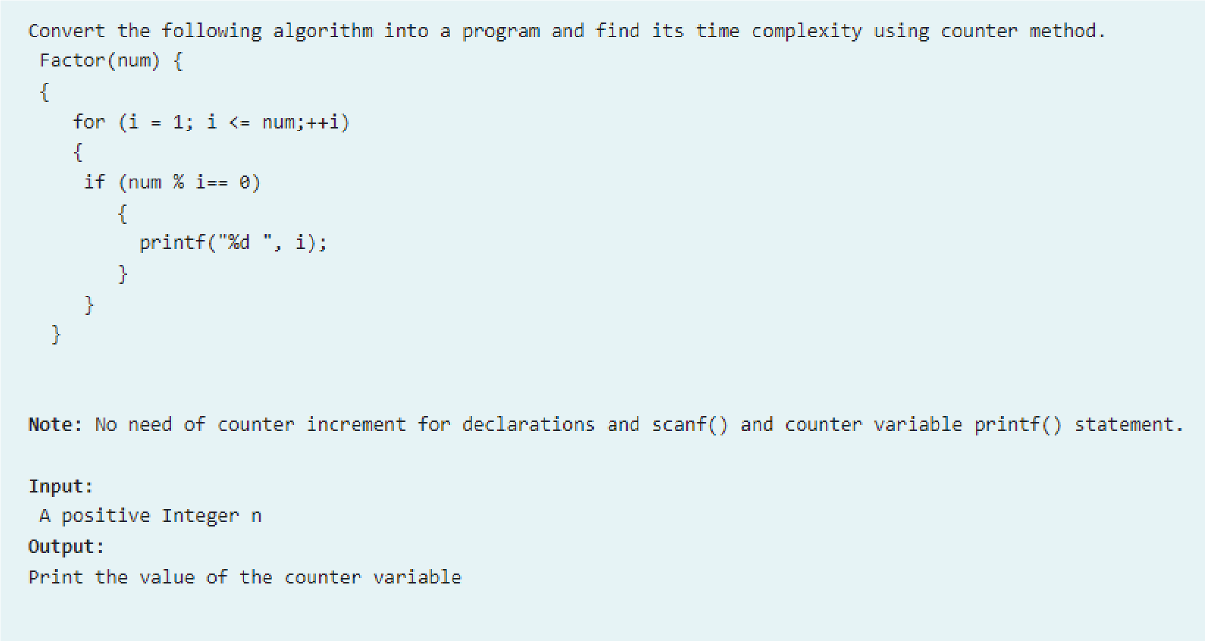
Marks for this submission: 1.00/1.00.

RESULT:

Thus the above code is executed successfully and gives the expected output.

PROBLEM 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);
        }
     }
  }


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:

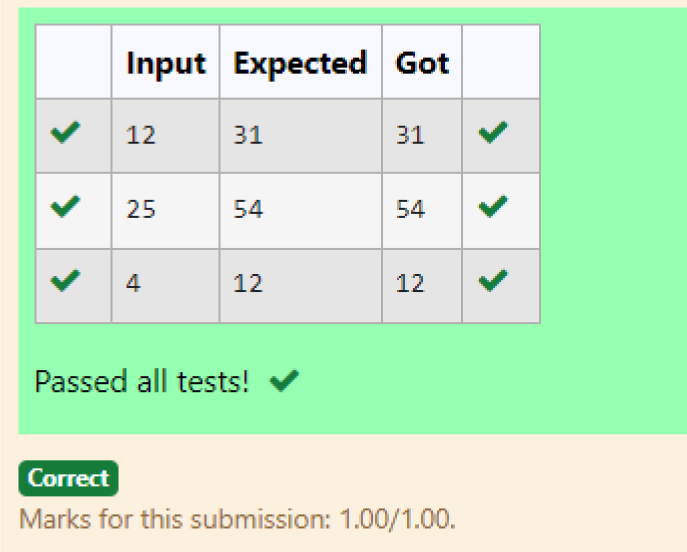
```
1. Input num:
  - Read an integer num from the user.
2. Initialize count = 0.
3. Iterate from i = 1 to num:
  - For each iteration:
    - Increment count twice (loop start and increment).
    - Check if num % i == 0:
      - If true, increment count.
4. After the loop, increment count once.
5. Print the value of count.
6. End.
```

CODE:

```
#include<stdio.h>
int main()
{
   int num,count=0;
```

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

OUTPUT:



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

Passed all tests! ✓

Correct

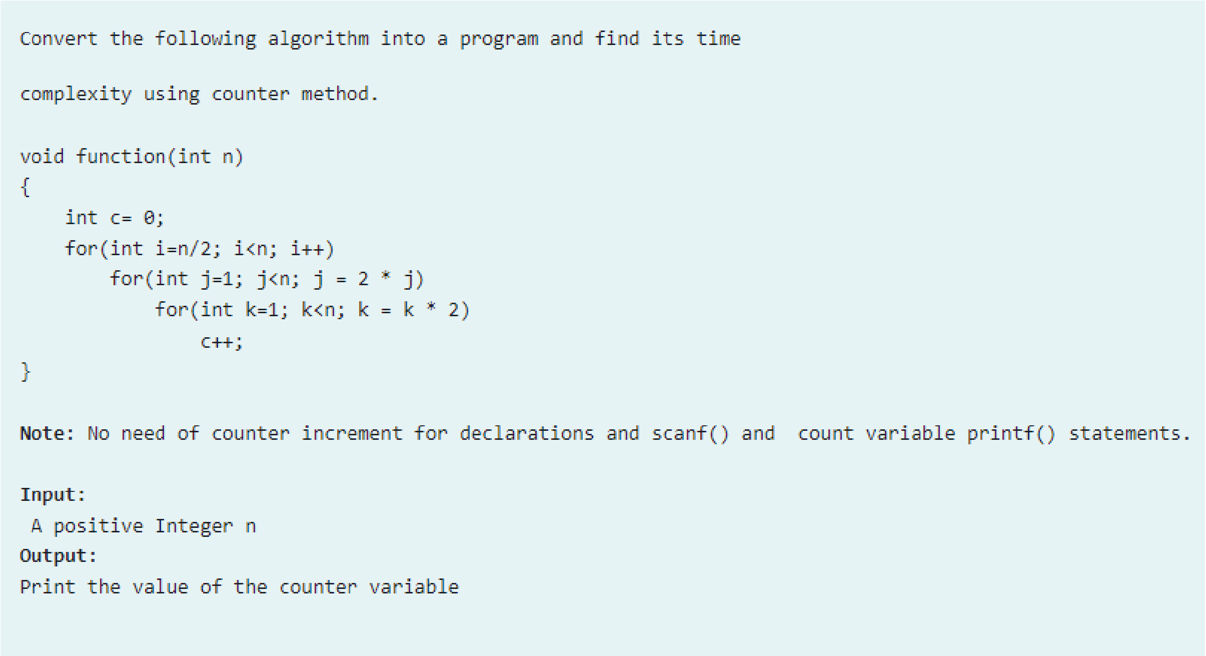
Marks for this submission: 1.00/1.00.

RESULT:

Thus the above code is executed successfully and gives the expected output.

PROBLEM 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++;
}
```

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:

```
1. Input n.
2. Initialize count = 0 and c = 0. Increment count.
3. For i from n/2 to n-1:
  - Increment count.
  - For j starting from 1, doubling until n-1:
    - Increment count.
    - For k starting from 1, doubling until n-1:
      - Increment count and c.
  - Increment count after the inner loop.
4. Increment count after each outer loop.
5. Increment count once after all loops.
6. Print count.
7. End.
```

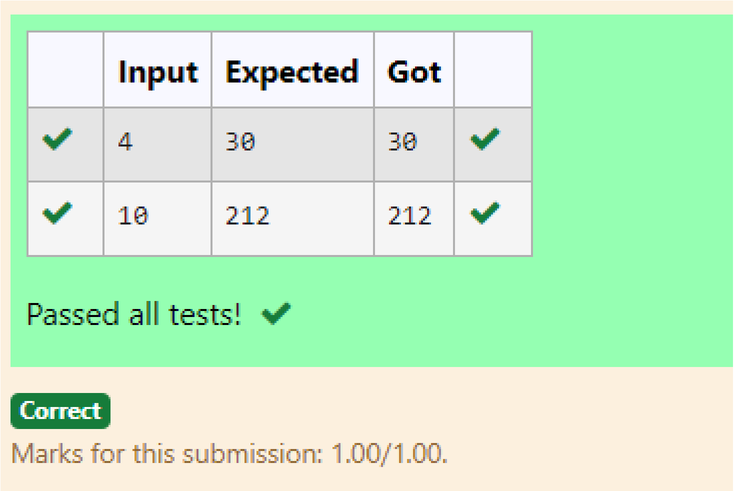
CODE:

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

```
{
    int n,count = 0;
    scanf("%d",&n);
    int c = 0;
    count++;
    for(int i = n/2;i < n;i++)
    {
        count++;
        for(int j = 1;j < n;j = 2*j)
        {
            count++;
            for(int k = 1;k < n;k = k*2)
            {
                count++;
                c++;
                count++;

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

OUTPUT:



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

Passed all tests! ✓

Correct

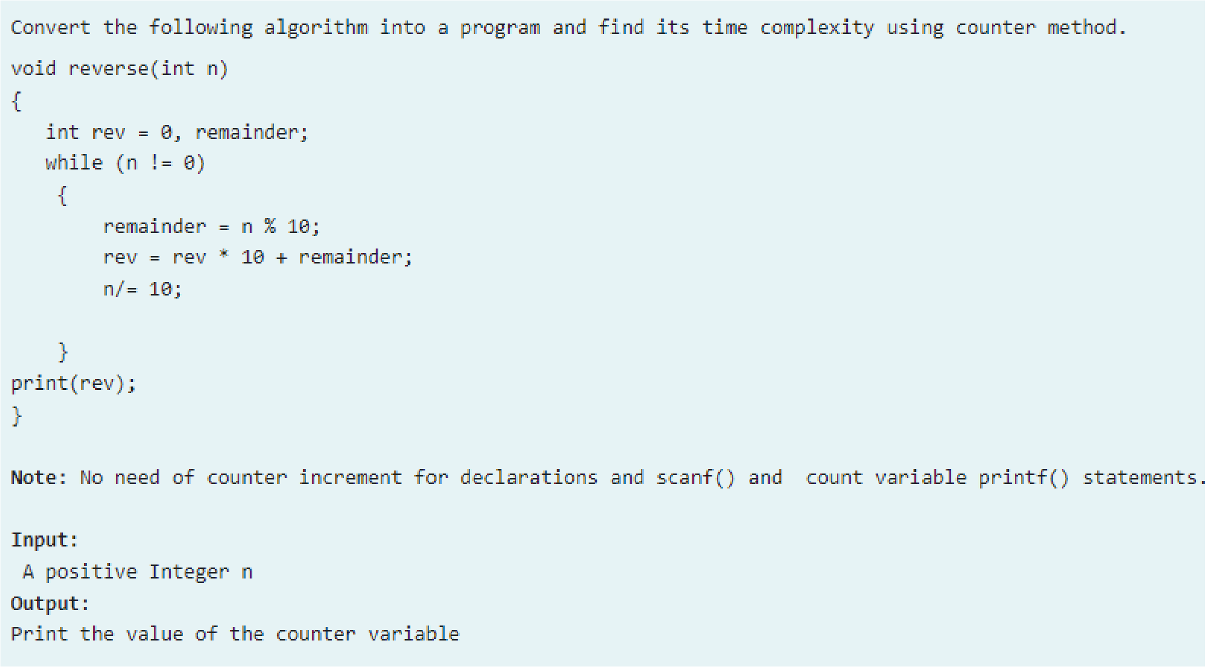
Marks for this submission: 1.00/1.00.

RESULT:

Thus the above code is executed successfully and gives the expected output.

PROBLEM 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);
}
```

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:

1. Input n and initialize rev = 0, count = 0, and remainder. Increment count.

2. While n != 0:

   - Increment count.

- Compute remainder = n % 10, update rev = rev * 10 + remainder, and update n = n / 10, incrementing count for each operation.

3. Increment count for the final loop condition and two additional operations.

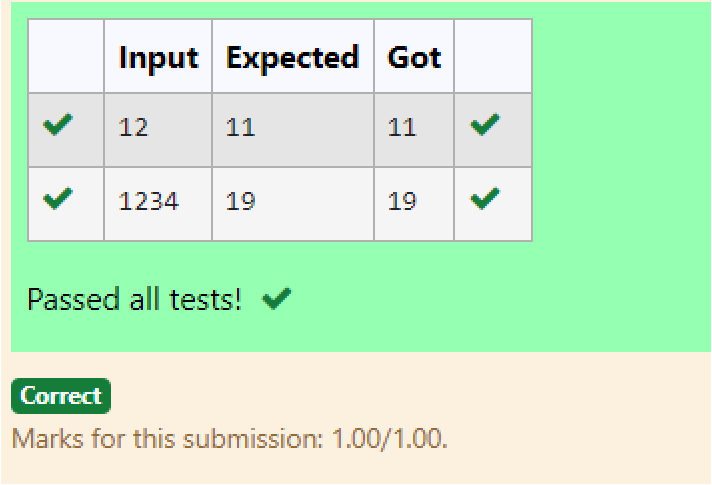
4. Print count.

5. End.

CODE:

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

OUTPUT:



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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

RESULT:

Thus the above code is executed successfully and gives the expected output.