Ex. No: 2 **Date: 20.08.24**

Register No.: 230701374 **Name: Velan A**

Finding Time Complexity of Algorithms

2.a. Finding Complexity using Counter Method

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 printf() statements. count variable

Input:
A positive Integer n

Output:
Print the value of the counter variable

Program:

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

```
void function(int n)
{
    int counter = 0;
    int i = 1;
    counter++;
    int s = 1;
    counter++;
    while(s<=n)
    {
        i++;
        s += i;
        counter++;
        counter++;
        counter++;
    }
    counter++;
    printf("%d",counter);
}

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

Output:

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

Passed all tests! ✔

2.b. Finding Complexity using Counter Method

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 printf() statements. count variable

Input:

A positive Integer n

Output:

Print the value of the counter variable

Program:

```
#include <stdio.h>

void func(int n)

{

   int counter = 0;

   if(n==1)
```

```
    {
     //printf("*");
    }
    else
    {
        for(int i=1;i<=n;i++)
        {
            counter++;
            for(int j=1;j<=n;j++)
            {
                counter++;
                counter++;
                break;
                counter++;
            }
            counter++;
            counter++;
        }
        counter++;
    }
    counter++;
    printf("%d",counter);
}

int main()
{
    int num;
```

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

Output:

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

2.c. Finding Complexity using Counter Method

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

Program :

```
#include <stdio.h>

void Factor(int num)

{

   int counter = 0;

   for(int i = 1;i <=num;i++)

   {

      counter++;

      if(num%i==0)

      {
```

```
            counter++;
            //printf("%d",i);
        }
        counter++;
    }
    counter++;
    printf("%d",counter);
}
int main()
{
    int n;
    scanf("%d",&n);
    Factor(n);
}
```

Output:

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

2.d. Finding Complexity using Counter Method

Aim: Convert the following algorithm into a program and find its timecomplexity 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 printf() statements. count variable

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

Program:

```
#include <stdio.h>

#include <stdlib.h>


void function(int n)

{

   int c=0;

   c++;

   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++;
           c++;
        }
        c++;
        c++;
      }
      c++;
      c++;
   }
   c++;
   printf("%d",c);
}
int main()
{
   int num;
   scanf("%d",&num);
   function(num);
}
```

Output:

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

2.e. Finding Complexity using Counter Method

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 printf() statements. count variable

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

Program:

```
#include <stdio.h>

void reverse(int n)

{

   int c=0;

   int rev = 0,rem;

   c++;

   c++;

   while(n!=0)

   {

      rem = n%10;
```

```
        rev = rev * 10 + rem;
        n/=10;
        c++;
        c++;
        c++;
        c++;

    }
    c++;
    //printf("%d",rev);
    printf("%d",c);
}
int main()
{
    int num;
    scanf("%d",&num);
    reverse(num);
}
```

Output:

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