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

Register No.: 230701360 **Name: Tejushree sanjeevikumar**

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

Algorithm:

void function(int n){

set count = 0

set i = 1

increment count by 1

```
    set s = 1
    increment count by 1

    while (s <= n){
        increment count by 1
        increment i by 1
        increment count by 1
        set s = s + i
        increment count by 1
    }
    increment count by 1
    print count
}
```

Program:

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

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

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

Algorithm:

void func(int n){

initialize count to 0

if n = 1{

increment count by 1

print "*"

}

else{

increment count by 1

```
    // outer loop from 1 to n
    for each i from 1 to n{
      increment count by 1

      // inner loop from 1 to n
      for each j from 1 to n {
        increment count by 1

        // simulate print statements with count increments
        increment count by 1  // first simulated printf("*")
        increment count by 1  // second simulated printf("*")

        // exit inner loop immediately
        increment count by 1  // break statement
      }
      increment count by 1
    }
    increment count by 1
  }
  print count
}
```

Program:

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

```
 scanf("%d",&n);
 if(n==1)
 {
  count++;
  //printf("*");
 }
 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);
}
```

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

Algorithm:

void function(int n){

set count = 0

increment count by 1

read n from input

increment count by 1

for i = 1 to n:

increment count by 1

if (n % i == 0):

increment count by 1

increment count by 1

```
    increment count by 1
    print count
}
```

Program:

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

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

Algorithm:

void function(int n){

set count = 0

increment count by 1

read n from input

increment count by 1

set c = 0

increment count by 1

for i = n / 2 to n - 1:

increment count by 1

for j = 1 to n (j = 2 * j each iteration):

```
            increment count by 1
            for k = 1 to n (k = 2 * k each iteration):
                increment count by 1
                increment c by 1
                increment count by 1
            increment count by 1
        increment count by 1
    increment count by 1


    increment count by 1
    print count
}
```

Program:

```
#include<stdio.h>
int main()
{
    int count=0;
    int n;
    count++;
    scanf("%d",&n);
    int c= 0;
    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	✔

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

Algorithm:

void function(int n){

set count = 0

increment count by 1

read n from input

increment count by 1

set rev = 0

set remainder = 0

increment count by 1

while n != 0:

```
        increment count by 1
        set remainder = n % 10
        increment count by 1
        set rev = rev * 10 + remainder
        increment count by 1
        set n = n / 10
        increment count by 1

    increment count by 1
    print count
}
```

Program:

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

```
        n/= 10;
        count++;

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

}
```

Output:

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