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

Register No.: 230701357 **Name: Swetha J**

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>

void function(int
n){     int count=0;
int i=1;     count++;
int s=1;     count++;
while(s<=n){         c
ount++;         i++;
count++;
s+=i;

       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! ✔

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 count variable printf() statements.

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>
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	✔

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:

function Factor(num)

{ initialize count to 0

// loop from 1 to num

for each i from 1 to num

{ increment count by 1

// check if i is a factor of num

if num modulo i equals 0

{ increment count by 1

// simulate printing i (e.g., printf("%d ", i);)

```
        }

        increment count by 1  // end of inner if-statement
    }

    increment count by 1  // after loop completion

    print count
}
```

Program:

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

```
}

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

Algorithm:

function(n)

{ initialize count to

0 initialize c to 0

increment count by 1

// outer loop: i goes from n/2 to n-1

for each i from n/2 to n-1

{ increment count by 1

// middle loop: j starts at 1 and doubles each iteration until j < n

for each j starting from 1 and doubling each time (j = 2 * j) until j < n

{ increment count by 1

// inner loop: k starts at 1 and doubles each iteration until k < n

for each k starting from 1 and doubling each time (k = k * 2) until k < n

{ increment count by 1 increment c by 1 increment

count by 1

}

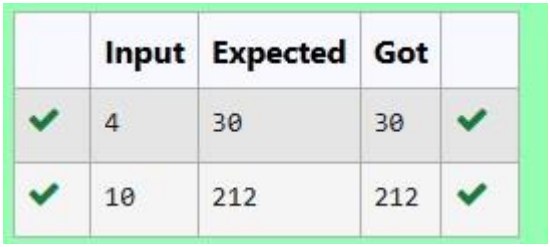
```
            increment count by 1  // after inner loop ends
        }

        increment count by 1  // after middle loop ends
    }

    increment count by 1  // after outer loop ends

    print count
}
```

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){                 count++;
c++;                 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	✔

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:

function reverse(n)

{ initialize count to 0

initialize rev to 0 initialize

remainder

increment count by 1 // for initialization

// loop until n is not equal to 0 while

n is not equal to 0 { increment count

by 1 // start of loop

remainder = n modulo 10

increment count by 1 // after calculating remainder

rev = rev * 10 + remainder increment

count by 1 // after updating rev

n = n divided by 10 increment

count by 1 // after updating n

}

increment count by 1 // after loop ends

```
    // simulate printing rev (e.g., print(rev))
increment count by 1  // for print statement


    print count
}
```

Program:

```
#include<stdio.h> void
reverse(int n)
{
   int count=0;     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++;
printf("%d",count);
}
```

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

Output:

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