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
    }
   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	✔