**FINDING TIME COMPLEXITY OF THE ALGORITHMS**

**AIM:**

Finding Complexity using Counter Method.

**ALGORITHM:**

1. Read integer `n`, set `count` to 2, `i` to 1, and `s` to 1.

2. While `s <= n`, increment `count`, update `i` (increment by 1), and update `s` (add `i` to `s`).

3. After exiting the loop, increment `count` by 1.

4. Print the value of `count`.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

int main()

{

int n,count=0,i=1,s=1;

scanf("%d",&n);

count++;

count++;

while (s<=n)

{

count++;

i++;

count++;

s=s+i;

count++;

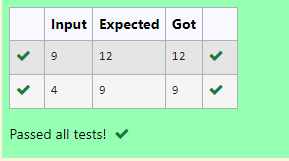
}

count++;

printf("%d",count);

}

**OUTPUT:**

****

**AIM:**

Finding Complexity using Counter Method.

**ALGORITHM:**

1. Read integer `n` and initialize `count` to 0.

2. If `n` is 1:

- Increment `count`.

- Print "\*".

- Increment `count`.

3. Else:

- Increment `count`.

- For each `i` from 1 to `n`:

- Increment `count`.

- Execute a nested loop:

- Increment `count` three times.

- Break the nested loop.

- Increment `count`.

4. Print the value of `count`.

**PROBLEM:**

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

**PROGRAM:**

#include<stdio.h>

int main()

{

int n,count=0;

scanf("%d",&n);

if(n==1)

{

count++;

printf("\*");

count++;

}

else

{

count++;

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

{

count++;

for(int j=1; j<=n; j++)

{

count++;

count++;

count++;

break;

}

count++;

}

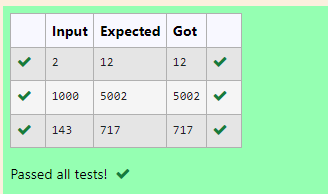
count++;

}

printf("%d",count);

}

**OUTPUT:**

****

**AIM:**

Finding Complexity using Counter Method.

**ALGORITHM:**

1. Read integer `num` and initialize `count` to 0.

2. For each integer `i` from 1 to `num`:

- Increment `count` twice.

- If `num % i == 0`, increment `count`.

3. Increment `count` once more after the loop.

4. Print the value of `count`.

**PROBLEM:**

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>

int main()

{

int num,count=0;

scanf("%d",&num);

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

{

count++;

count++;

if (num % i== 0)

{

count++;

}

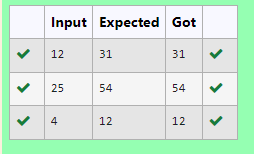
}

count++;

printf("%d",count);

}

**OUTPUT:**

****

**AIM:**

Finding Complexity using Counter Method.

**ALGORITHM:**

1. Read integer `n` and initialize `count` to 0 and `c` to 0.

2. Increment `count` once.

3. For each integer `i` from `n/2` to `n - 1`:

- Increment `count`.

- For each integer `j` starting from 1, doubling each time until it is less than `n`:

- Increment `count`.

- For each integer `k` starting from 1, doubling each time until it is less than `n`:

- Increment `count`.

- Increment `c`.

- Increment `count`.

- Increment `count`.

- Increment `count`.

4. Increment `count` once more after the outer loop.

5. Print the value of `count`.

**PROBLEM:**

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

**PROGRAM:**

#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++;

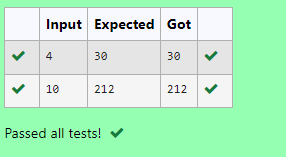
}

count++;

printf("%d",count);

}

**OUTPUT:**

****

**AIM:**

Finding Complexity using Counter Method.

**ALGORITHM:**

1. Read integer `n`, initialize `rev` to 0 and `count` to 0.

2. Increment `count`.

3. While `n` is not equal to 0:

- Increment `count`.

- Set `remainder` to `n % 10`.

- Increment `count`.

- Update `rev` to `rev \* 10 + remainder`.

- Increment `count`.

- Update `n` to `n / 10`.

- Increment `count`.

4. Increment `count` twice after the loop.

5. Print the value of `count`.

**PROBLEM:**

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

**PROGRAM:**

#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++;

}

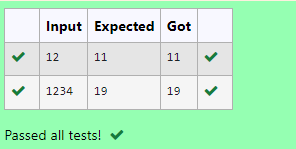
count++;

count++;

printf("%d",count);

}

**OUTPUT:**

****