Finding Complexity using Counter Method

PROGRAM 1:
AIM: Finding Complexity using Counter Method

ALGORITHM:
1. Initialize `counter = 0` and `i = 1`.
2. Increment `counter` after initializing `i` and `s`.
3. Read the integer `n`.
4. While `s <= n`, increment `counter`, then increment `i`.
5. Add `i` to `s` and increment `counter` for each iteration.
6. After exiting the loop, increment `counter` and print its value.

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.

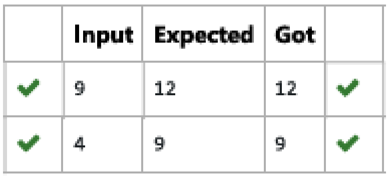
Input:
 A positive Integer n
Output:
Print the value of the counter variable

PROGRAM:

```
#include <stdio.h>
int main()
{
   int counter=0;
   int i=1;
   counter++;
   int s=1;
```

```
    counter++;
    int n;
    scanf("%d",&n);
    while (s<=n)
    {
        counter++;
        i++;
        counter++;
        s=s+i;
        counter++;
    }
    counter++;
    printf("%d",counter);
}
```

OUTPUT:



	Input	Expected	Got	
✓	9	12	12	✓
✓	4	9	9	✓

PROBLEM 2:

AIM: Finding Complexity using Counter Method

ALGORITHM:

1. Read integer `n` and initialize `counter = 0`.

2. If `n == 1`, increment `counter` and exit.

3. For `n > 1`, increment `counter` for the outer loop, then for each inner loop iteration:

- Increment `counter` and break after one iteration.

4. Print the final `counter` value.

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

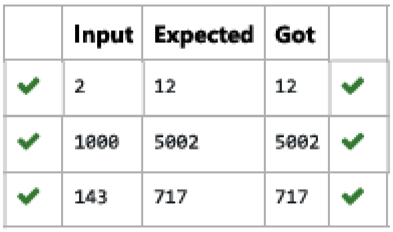
Output:

Print the value of the counter variable

PROGRAM:

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

    if (n == 1) {
        counter++;
        //printf("*");
    } else {
        counter++;
        for (int i = 1; i <= n; i++)
        {
            counter++;

            for (int j = 1; j <= n; j++) {
                counter++;
                //printf("*");
                counter++;
                //printf("*");
                counter++;

                break;
            }
            counter++;
        }
        counter++;
    }
    printf("%d\n", counter);
    return 0;
}
```

OUTPUT:



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

PROBLEM 3:

AIM: Finding Complexity using Counter Method

ALGORITHM:

```
1. Read integer `num` and initialize `counter = 0`.
2. Loop from `i = 1` to `i = num`:
   - Increment `counter` twice for each iteration.
   - If `num % i == 0`, increment `counter`.
3. Increment `counter` once after the loop.
4. Print the final value of `counter`.
```

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);
        }
     }
  }
```

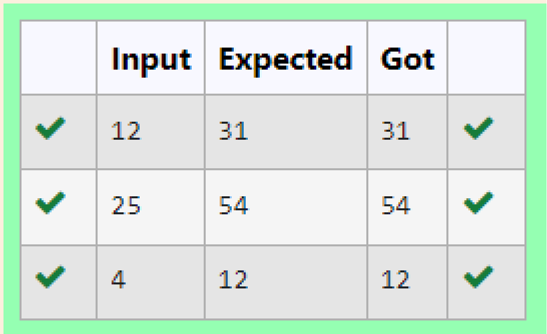
Note: No need of counter increment for declarations and scanf() and counter variable printf() statement.

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

PROGRAM:

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

    }
    counter++;
    printf("%d",counter);
}
```

OUTPUT:



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

PROBLEM 4:

AIM: Finding Complexity using Counter Method

ALGORITHM:

1. Read integer `n` and initialize `counter = 0`.

2. Initialize `c = 0` and increment `counter`.

3. Loop `i` from `n/2` to `n-1`:

   - Increment `counter`.

4. Inside the outer loop, loop `j` from 1, doubling each time (`j = 2 * j`), until `j < n`:

   - Increment `counter`.

5. Inside the middle loop, loop `k` from 1, doubling each time (`k = 2 * k`), until `k < n`:

   - Increment `counter` and `c`.

6. Increment `counter` after each inner loop and outer loop iteration.

7. Print the final value of `counter`.

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

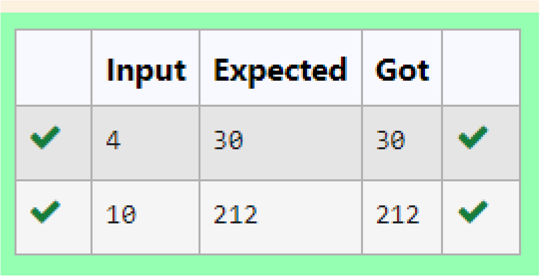
Output:

Print the value of the counter variable

PROGRAM:

```
#include <stdio.h>
int main()
{
    int n,counter=0;
    scanf("%d",&n);
    int c= 0;
    counter++;
    for(int i=n/2; i<n; i++)
    {
        counter++;
        for(int j=1; j<n; j = 2 * j)
        {
            counter++;
            for(int k=1; k<n; k = k * 2)
            {
                counter++;
                c++;
                counter++;
            }
            counter++;
        }
        counter++;
    }
    counter++;
    printf("%d",counter);
}
```

OUTPUT:



	Input	Expected	Got	
✓	4	30	30	✓
✓	10	212	212	✓

PROBLEM 5:

AIM: Finding Complexity using Counter Method

ALGORITHM:

1. Read integer `n` and initialize `rev = 0`, `counter = 0`, and `remainder`.

2. Increment `counter` before starting the loop.

3. While `n != 0`:

   - Increment `counter`.

   - Calculate `remainder = n % 10` and increment `counter`.

   - Update `rev = rev * 10 + remainder` and increment `counter`.

   - Update `n /= 10` and increment `counter`.

4. After the loop, increment `counter` twice.

5. Print the final value of `counter`.

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

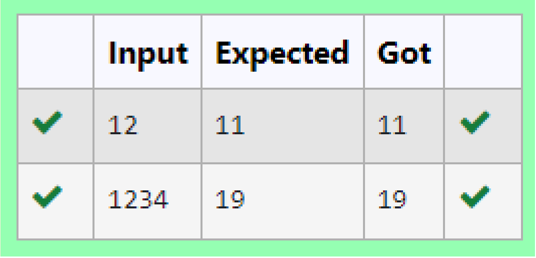
Output:

`Print the value of the counter variable`

PROGRAM:

```
#include <stdio.h>

int main()

{

    int n,rev = 0,counter=0,remainder;

    counter++;

    scanf("%d",&n);

    while (n != 0)

    {

        counter++;
        remainder = n % 10;
        counter++;
        rev = rev * 10 + remainder;
        counter++;
        n/= 10;
        counter++;
    }
    counter++;
    counter++;
    printf("%d",counter);
}
```

OUTPUT:



	Input	Expected	Got	
✓	12	11	11	✓
✓	1234	19	19	✓