Finding Time Complexity of Algorithms

1.

```
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

Algorithm:

1. Start.
2. Input integer a.
3. Initialize i = 1, s = 1, c = 1.
4. While s <= a:
5. Increment i by 1.
6. Increment c by 1.
7. Add i to s.
8. Increment c by 2.
9. Output value of c.
10. End.

Program:

```
#include<stdio.h>

void function(int n){

   int i=1;int c=1;

   int s=1;c++;
```

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

	Input	Expected	Got	
✔	9	12	12	✔
✔	4	9	9	✔

2.

```
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:

Program:

```
#include<stdio.h>
void func(int n)
{       int count=0;
        count++;
        if(n==1)
        {
```

```
        }
        else
        {
                for(int i=1; i<=n; i++)
                {
                        for(int j=1; j<=n; j++)
                        {       count++;
                                count++;
                                count++;
                                count++;
                                break;
                        }
                        count++;
                }
                count++;
        }
        printf("%d",count);
}
int main() {
        int a;
        scanf("%d",&a);
        func(a);
}
```

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

3.

```
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:

1. Start.
2. Input integer a.
3. Initialize count = 0.
4. Increment count by 1.
5. If n == 1:
6. Do nothing.
7. Else:
8. For i = 1 to n:
9. Increment count by 1.
10. Output value of count.
11. End.

Program:

#include<stdio.h>

```
void Factor(int num) {
	int c=0;
	for(int i=1; i<=num; ++i) {
		c++;
		c++;
		if(num%i==0) {

			c++;
		}
	}
	c++;
	printf("%d",c);
}
int main() {
	int a;
	scanf("%d",&a);
	Factor(a);
}
```

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

4.

```
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
```

Algorithm:

1. Start.
2. Input integer a.
3. Initialize count = 1, c = 0.
4. For i = n/2 to n-1:
5. Increment count by 1.
6. For j = 1 to n (with j = 2 * j):
7. Increment count by 1.
8. Output value of count.
9. End.

Program:

```
#include<stdio.h>

void function(int n)

{

   int c= 0;int count=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++;
                count++;
                c++;
            }
            count++;
        }
        count++;
    }
    count++;
    printf("%d",count);
}
int main()
{
    int a;
    scanf("%d",&a);
    function(a);
}
```

	Input	Expected	Got	
✔	4	30	30	✔
✔	10	212	212	✔

5.

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

Algorithm:

1. Start.
2. Input integer num.
3. Initialize count = 1, rev = 0.
4. While n != 0:
5. Increment count by 1.
6. Set rem = n % 10, increment count by 1.
7. Update rev = rev * 10 + rem, increment count by 1.
8. Update n = n / 10, increment count by 1.
9. Increment count by 2.
10. Output value of count.
11. End

Program:

```
#include<stdio.h>

void reverse(int n)

{
```

```
    int count=0;
    int rev = 0, rem;
    count++;
    while (n != 0)
    {
        count++;
        rem = n % 10;count++;
        rev = rev * 10 + rem;count++;
        n/= 10;count++;

    }
    count++;
    count++;
    printf("%d",count);
}
int main()
{
    int num;
    scanf("%d",&num);
    reverse(num);
}
```

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