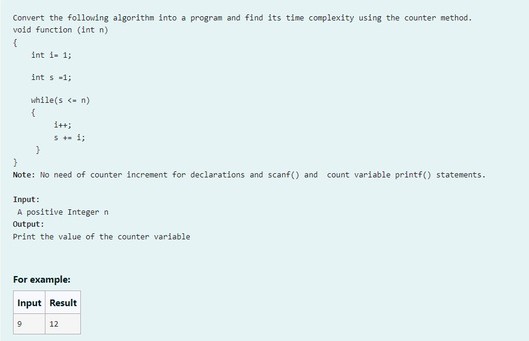
REG NO: 230701055 NAME : BHUVANESHWARI K

DEPT : CSE - A

TIME COMPLEXITY

# QUESTION 2.A

**AIM**:



# AIM:

Step 1: Start

Step 2: Input the integer n

Step 3: Initialize c to 0 to count operations Step 4: Initialize i to 1

Step 5: Increment c by 1

Step 6: Initialize s to 1

Step 7: Increment c by 1

Step 8: While s is less than or equal to n, do Steps 8.1 to 8.5

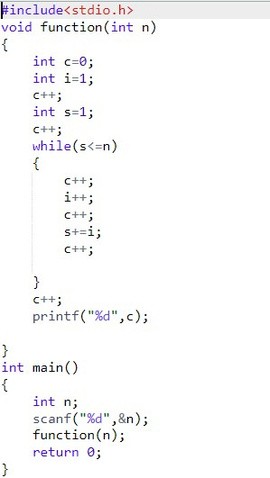
Step 8.1: Increment c by 1

Step 8.2: Increment i by 1

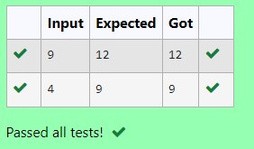
Step 8.3: Increment c by 1 Step 8.4: Add i to s (s += i) Step 8.5: Increment c by 1

Step 9: Increment c by 1 Step 10: Print the value of c Step 11: Stop

# PROGRAM:



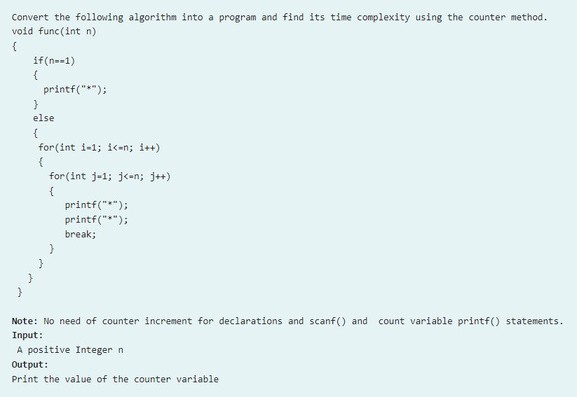
**OUTPUT :**



# RESULT:

The above code is executed successfully and gives expected output.

# QUESTION 2.b



**ALGORITHM:**

Step 1: Start

Step 2: Input the integer n

Step 3: Initialize c to 0 to count operations

Step 4: If n is equal to 1, go to Step 5, else go to Step 7 Step 5: Increment c by 1

Step 6: Print "\*" and go to Step 12

Step 7: Increment c by 1

Step 8: For each integer i from 1 to n, do Steps 9 to 11 Step 9: Increment c by 1

Step 10: For each integer j from 1 to n, do Steps 10.1 to 10.4

Step 10.1: Increment c by 1

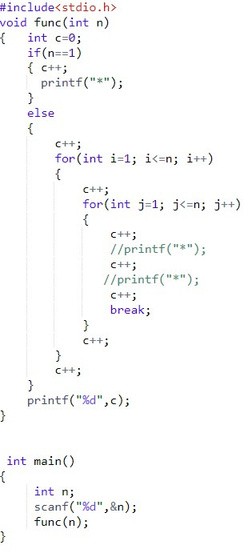
Step 10.2: Increment c by 1

Step 10.3: Increment c by 1

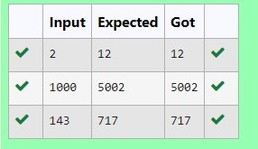
Step 10.4: Break out of the inner loop Step 11: Increment c by 1

Step 12: Increment c by 1 Step 13: Print the value of c Step 14: Stop

# PROGRAM:



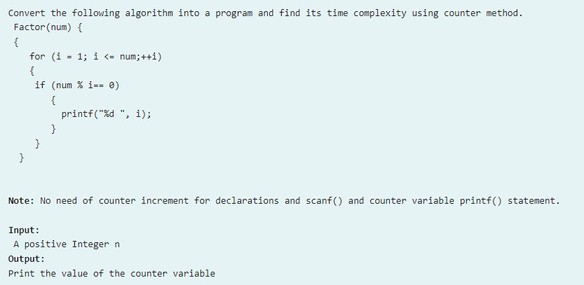
**OUTPUT:**



# RESULT:

The above code is executed successfully and gives expected output.

# QUESTION 2.C



**ALGORITHM:**

Step 1: Start

Step 2: Input the integer n

Step 3: Initialize c to 0 to count operations

Step 4: For each integer i from 1 to n, do Steps 5 to 7

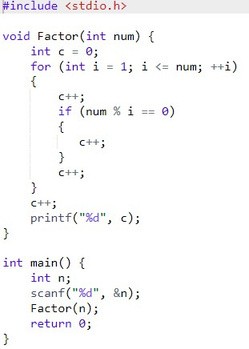
Step 5: Increment c by 1

Step 6: If n is divisible by i (n % i == 0), increment c by 1 Step 7: Increment c by 1

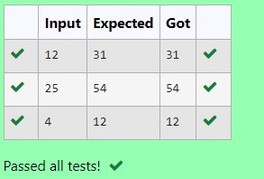
Step 8: Increment c by 1

Step 9: Print the value of c Step 10: Stop

# PROGRAM:



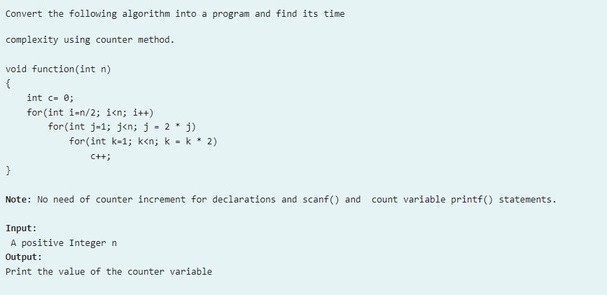
**OUTPUT:**



# RESULT:

The above code is executed successfully and gives expected output.

# QUESTION 2.D



**ALGORITM:**

Step 1: Start

Step 2: Input the integer n

Step 3: Initialize count to 0 to count operations Step 4: Initialize c to 0

Step 5: Increment count by 1

Step 6: For each integer i from n/2 to n - 1, do Steps 7 to 9 Step 7: Increment count by 1

Step 8: Initialize j to 1 and while j is less than n, do Steps 8.1 to 8.5 Step 8.1: Increment count by 1

Step 8.2: Initialize k to 1 and while k is less than n, do Steps 8.2.1 to 8.2.4 Step 8.2.1: Increment count by 1

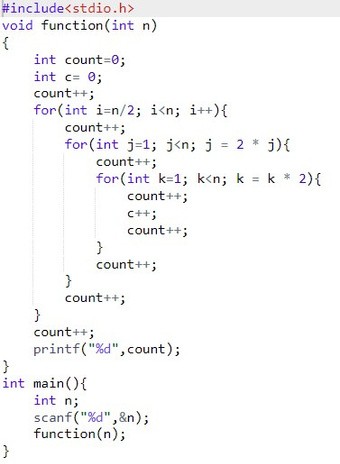
Step 8.2.2: Increment c by 1

Step 8.2.3: Increment count by 1 Step 8.2.4: Multiply k by 2 (k = k \* 2)

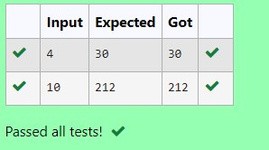
Step 8.3: Increment count by 1 Step 8.4: Multiply j by 2 ( j = j \* 2) Step 9: Increment count by 1

Step 10: Increment count by 1 Step 11: Print the value of count Step 12: Stop

# PROGRAM:



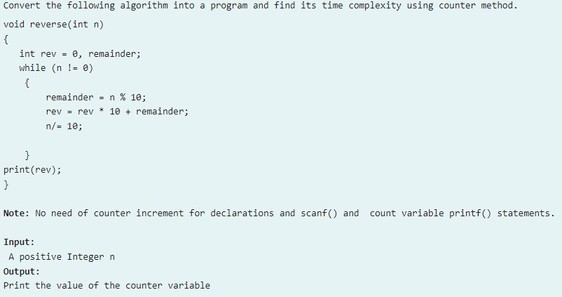
**OUTPUT:**



# RESULT:

The above code is executed successfully and gives expected output.

# QUESTION 2.E



**ALGORITHM:**

Step 1: Start

Step 2: Input the integer n

Step 3: Initialize counter to 0 to count operations Step 4: Initialize rev to 0 and remainder as unassigned Step 5: Increment counter by 1

Step 6: While n is not equal to 0, do Steps 6.1 to 6.7 Step 6.1: Increment counter by 1

Step 6.2: Calculate remainder as n % 10 Step 6.3: Increment counter by 1

Step 6.4: Update rev to rev \* 10 + remainder

Step 6.5: Increment counter by 1

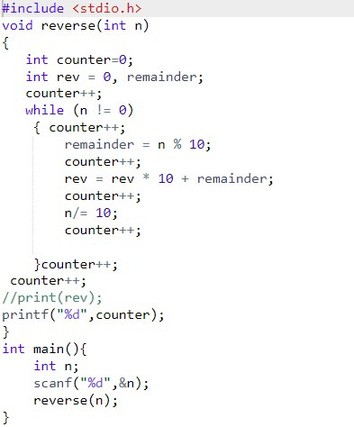
Step 6.6: Divide n by 10 (n /= 10)

Step 6.7: Increment counter by 1

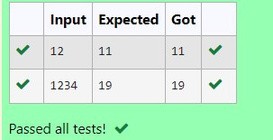
Step 7: Increment counter by 1

Step 8: Increment counter by 1 Step 9: Print the value of counter Step 10: Stop

# PROGRAM:



**OUTPUT:**



# RESULT:

The above code is executed successfully and gives expected output.