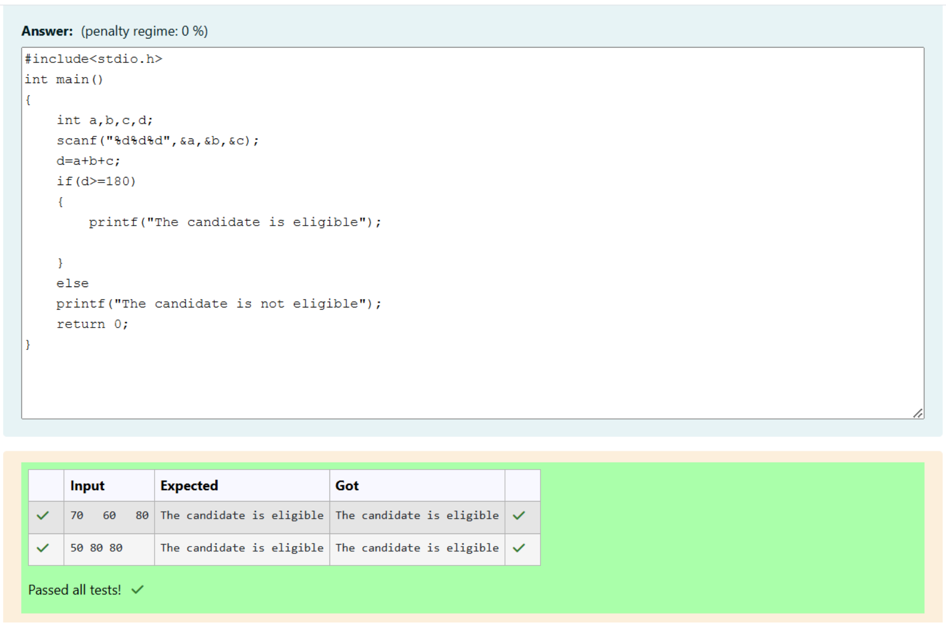
1.Write a C program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Maths >= 65

Marks in Physics >= 55

Marks in Chemistry >= 50



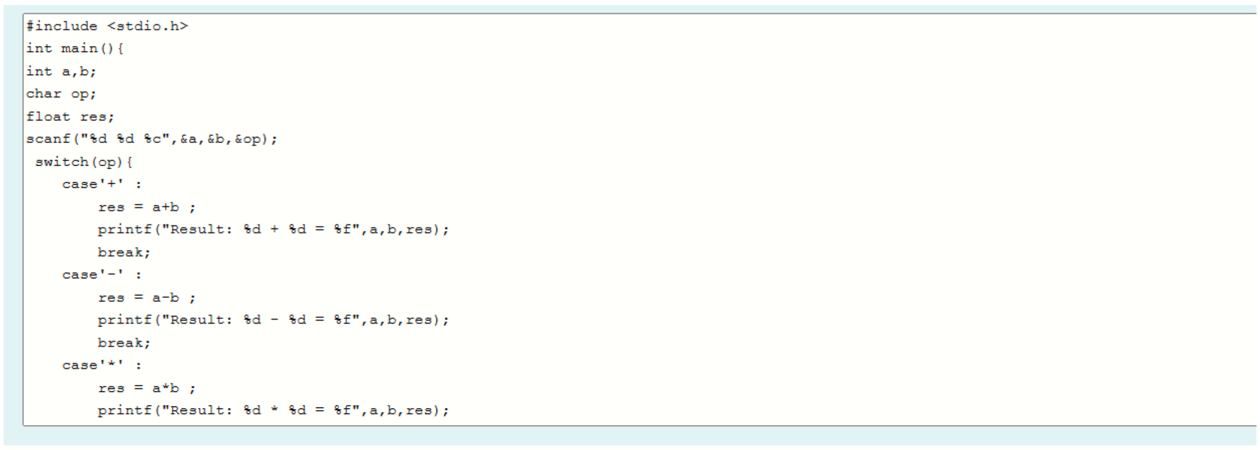
Answer: (penalty regime: 0 %)

```
#include<stdio.h>
int main()
{
    int a,b,c,d;
    scanf("%d%d%d",&a,&b,&c);
    d=a+b+c;
    if(d>=180)
    {
        printf("The candidate is eligible");

    }
    else
    printf("The candidate is not eligible");
    return 0;
}
```

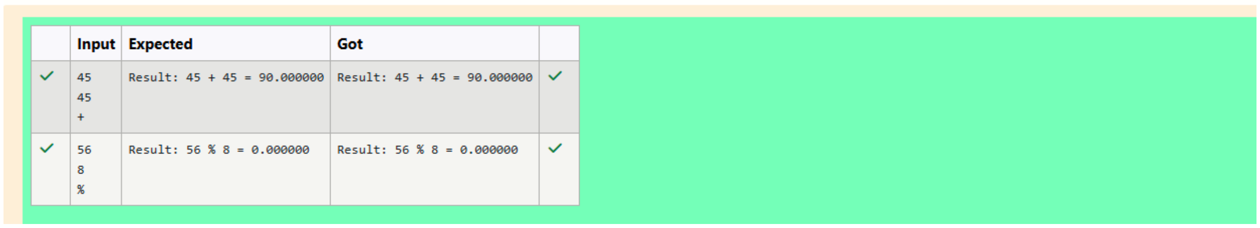
	Input	Expected	Got	
✓	70 60 80	The candidate is eligible	The candidate is eligible	✓
✓	50 80 80	The candidate is eligible	The candidate is eligible	✓

Passed all tests! ✓

2. Complete the calculator program with Basic operations (+, -, *, /, %) of two numbers using switch statement.



```
#include <stdio.h>
int main(){
int a,b;
char op;
float res;
scanf("%d %d %c",&a,&b,&op);
 switch(op){
    case'+' :
        res = a+b ;
        printf("Result: %d + %d = %f",a,b,res);
        break;
    case'-' :
        res = a-b ;
        printf("Result: %d - %d = %f",a,b,res);
        break;
    case'*' :
        res = a*b ;
        printf("Result: %d * %d = %f",a,b,res);
```

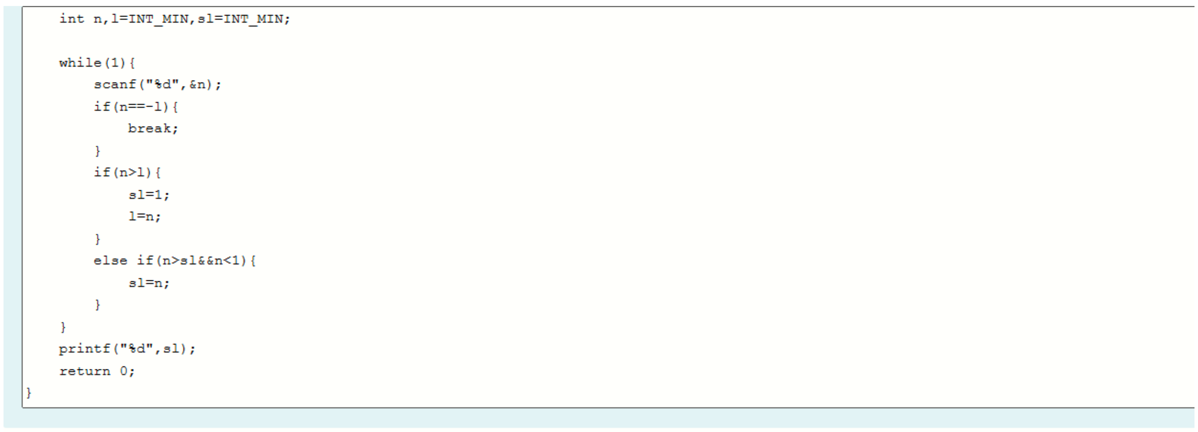


	Input	Expected	Got	
✓	45 45 +	Result: 45 + 45 = 90.000000	Result: 45 + 45 = 90.000000	✓
✓	56 8 %	Result: 56 % 8 = 0.000000	Result: 56 % 8 = 0.000000	✓

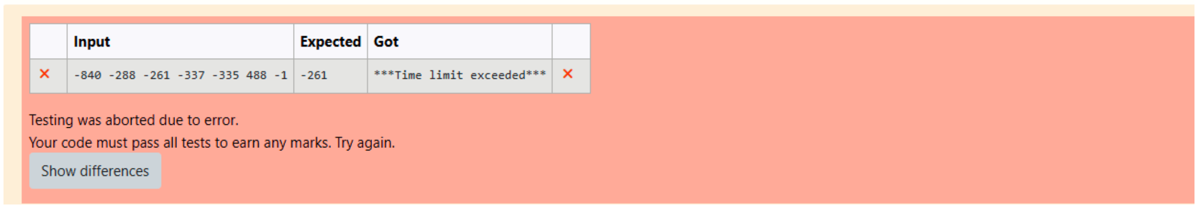
3. You are given a sequence of integers as input, terminated by a -1. (That is, the input integers may be positive, negative or 0. A -1 in the input signals the end of the input.)

-1 is not considered as part of the input.

Find the second largest number in the input. You may not use arrays.



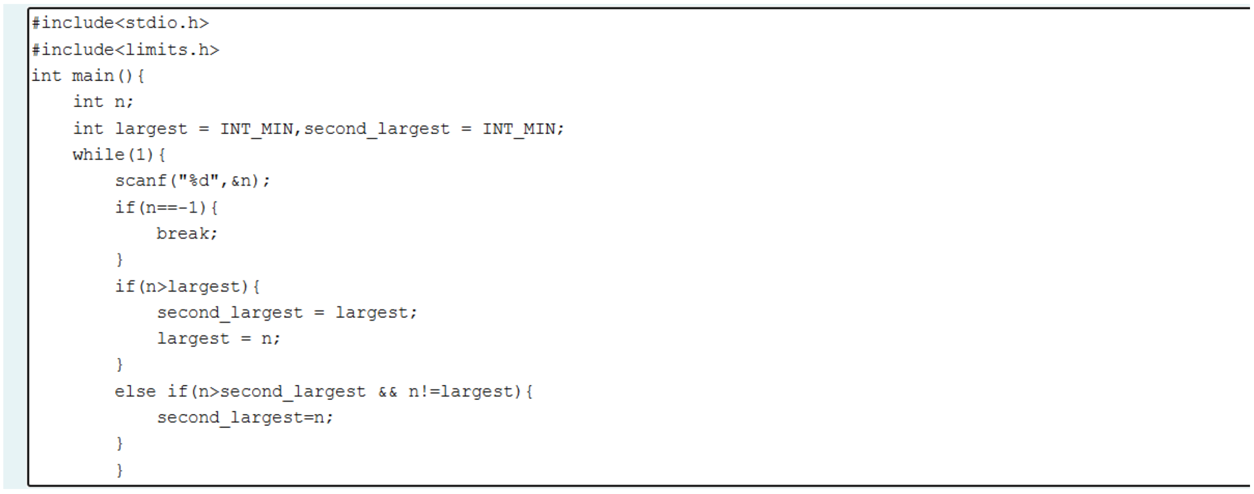
```
    int n,l=INT_MIN,sl=INT_MIN;

    while(1){
        scanf("%d",&n);
        if(n==-1){
            break;
        }
        if(n>l){
            sl=l;
            l=n;
        }
        else if(n>sl&&n<l){
            sl=n;
        }
    }
    printf("%d",sl);
    return 0;
}
```



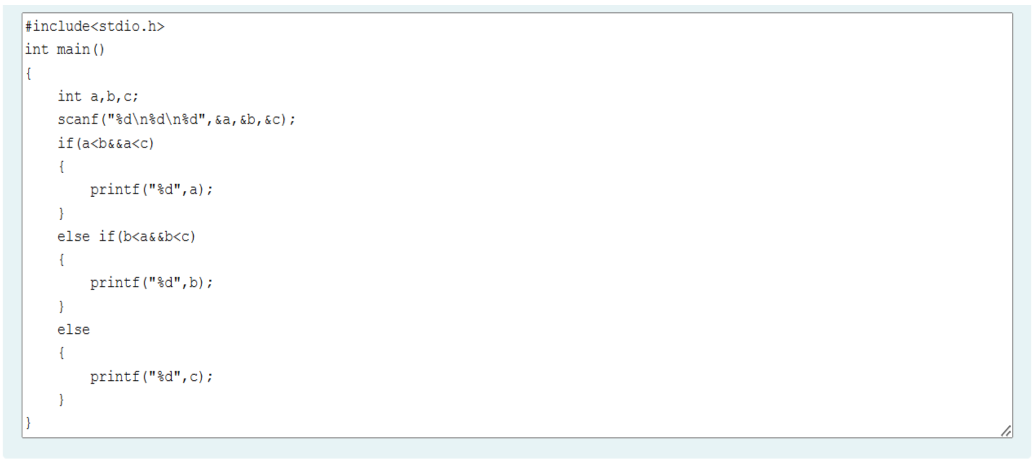
	Input	Expected	Got	
✗	-840 -288 -261 -337 -335 488 -1	-261	***Time limit exceeded***	✗

Testing was aborted due to error.
Your code must pass all tests to earn any marks. Try again.

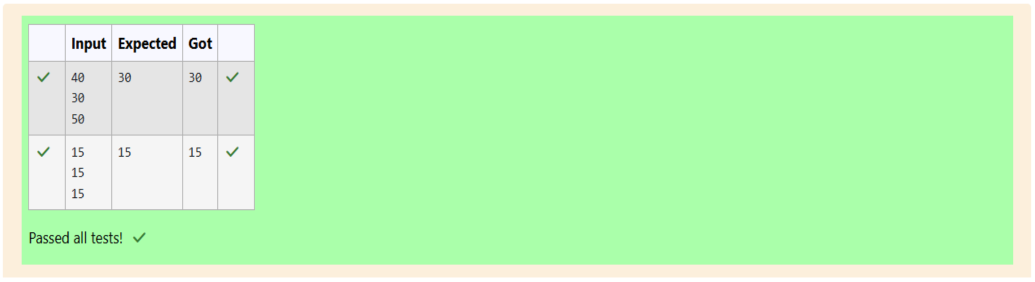
4. The lengths of the sides of a triangle X, Y and Z are passed as the input. The program must print the smallest side as the output.



```
#include<stdio.h>
#include<limits.h>
int main(){
    int n;
    int largest = INT_MIN,second_largest = INT_MIN;
    while(1){
        scanf("%d",&n);
        if(n==-1){
            break;
        }
        if(n>largest){
            second_largest = largest;
            largest = n;
        }
        else if(n>second_largest && n!=largest){
            second_largest=n;
        }
        }
```



```
#include<stdio.h>
int main()
{
    int a,b,c;
    scanf("%d\n%d\n%d",&a,&b,&c);
    if(a<b&&a<c)
    {
        printf("%d",a);
    }
    else if(b<a&&b<c)
    {
        printf("%d",b);
    }
    else
    {
        printf("%d",c);
    }
}
```



	Input	Expected	Got	
✓	40 30 50	30	30	✓
✓	15 15 15	15	15	✓

Passed all tests! ✓

5. #include <**stdio.h**>

int add(int, int);

int main()
{

int a = 10, b = 20;

printf("Sum of two numbers = %d\n", add(a, b)); // variables a, b are called actual arguments

return 0;
}

int add(**int** x, **int** y)
{

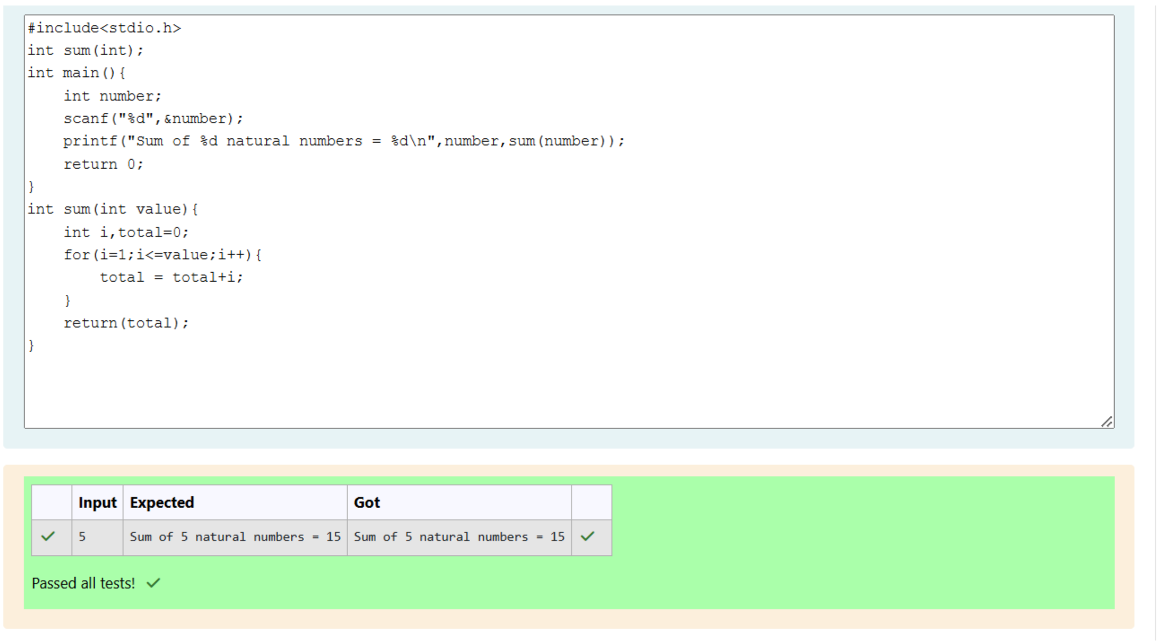
// variables x, y are called formal parameters
return(x + y);
}

In the above code whenever the function call add(a, b) is made, the execution control is transferred to the function definition of add().

The values of actual arguments a and b are copied in to the formal arguments x and y respectively.

The formal parameters x and y are available only with in the function definition of add(). After completion of execution of add(), the control is transferred back to the main().

See & retype the below code which will demonstrate about formal and actual arguments.



```
#include<stdio.h>
int sum(int);
int main(){
    int number;
    scanf("%d",&number);
    printf("Sum of %d natural numbers = %d\n",number,sum(number));
    return 0;
}
int sum(int value){
    int i,total=0;
    for(i=1;i<=value;i++){
        total = total+i;
    }
    return(total);
}
```

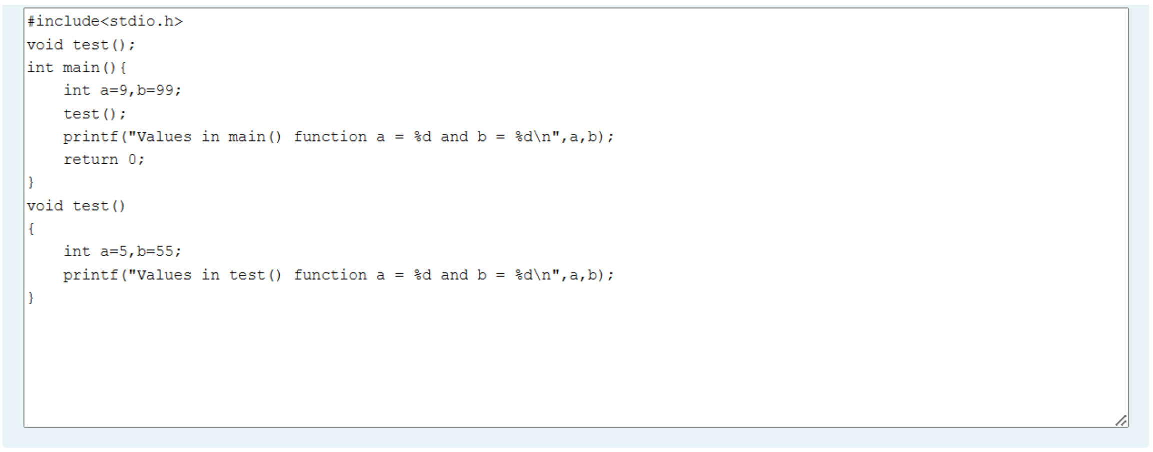
	Input	Expected	Got	
✓	5	Sum of 5 natural numbers = 15	Sum of 5 natural numbers = 15	✓

Passed all tests! ✓

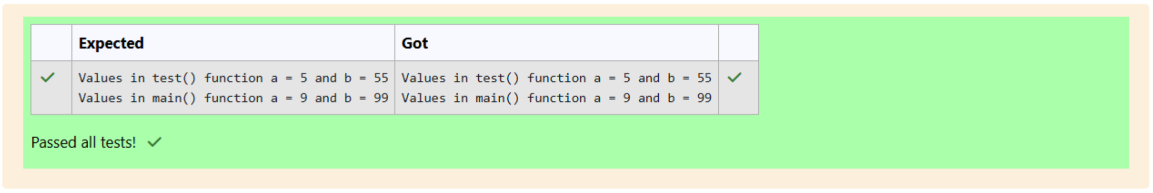
6. retype the below code which will demonstrate about local variables.

```
#include <stdio.h>

void test();

int main()
{
   int a = 9, b = 99;
   test();
   printf("Values in main() function a = %d and b = %d\n", a, b);
   return 0;
}

void test()
{
   int a = 5, b = 55;
   printf("Values in test() function a = %d and b = %d\n", a, b);
}
```

7. retype the below code which will demonstrate about global variables.



```
#include<stdio.h>
void test();
int main(){
    int a=9,b=99;
    test();
    printf("Values in main() function a = %d and b = %d\n",a,b);
    return 0;
}
void test()
{
    int a=5,b=55;
    printf("Values in test() function a = %d and b = %d\n",a,b);
}
```

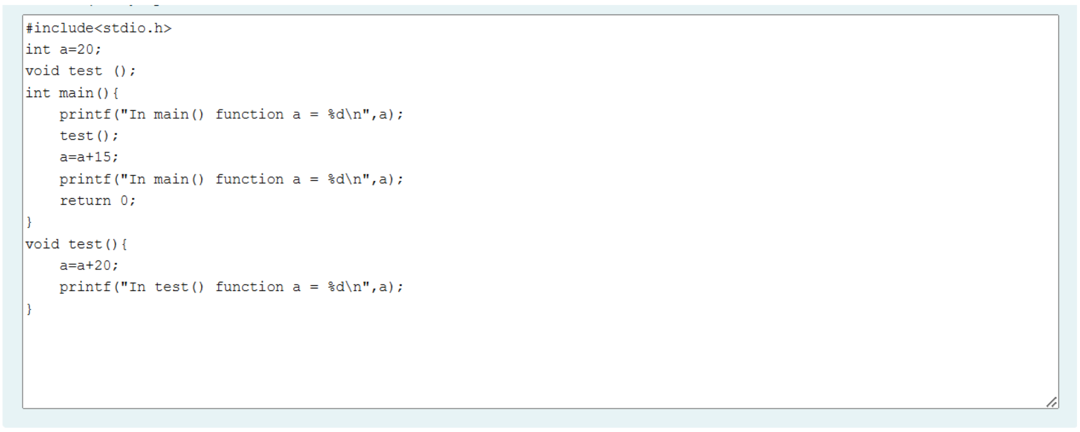


	Expected	Got	

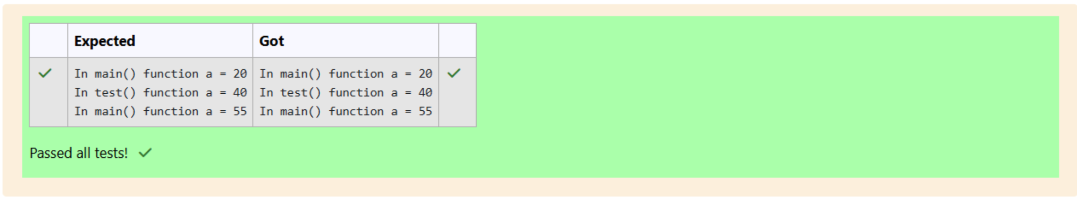
✓	`Values in test() function a = 5 and b = 55`
	`Values in main() function a = 9 and b = 99`
		`Values in test() function a = 5 and b = 55`
		`Values in main() function a = 9 and b = 99`
			✓

Passed all tests! ✓

```
#include <stdio.h>

int a = 20;

void test();

int main()
{
   printf("In main() function a = %d\n", a);
   test();
   a = a + 15;
   printf("In main() function a = %d\n", a);
   return 0;
}

void test()
{
   a = a + 20;
   printf("In test() function a = %d\n", a);
}
```



```
#include<stdio.h>
int a=20;
void test ();
int main(){
    printf("In main() function a = %d\n",a);
    test();
    a=a+15;
    printf("In main() function a = %d\n",a);
    return 0;
}
void test(){
    a=a+20;
    printf("In test() function a = %d\n",a);
}
```



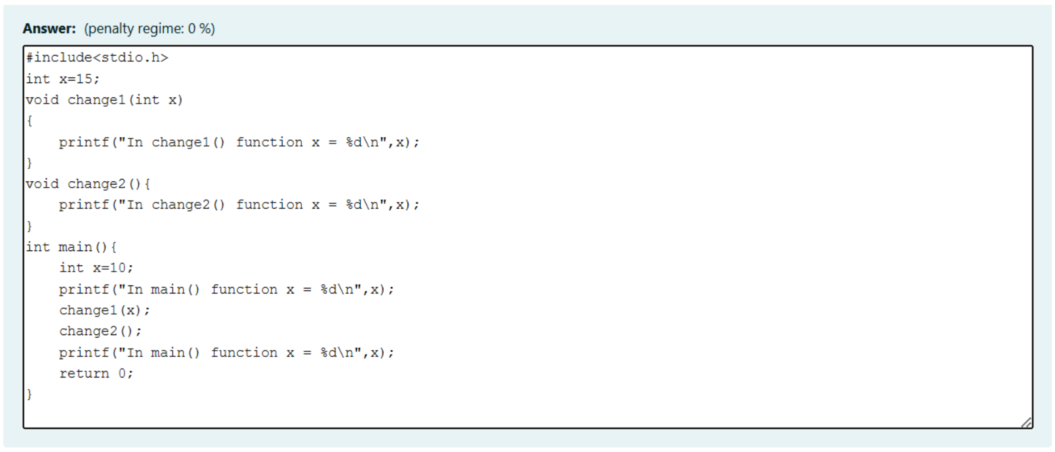
	Expected	Got	
✓	In main() function a = 20 In test() function a = 40 In main() function a = 55	In main() function a = 20 In test() function a = 40 In main() function a = 55	✓

Passed all tests! ✓

8. retype the below code which will demonstrate about local and global variables.

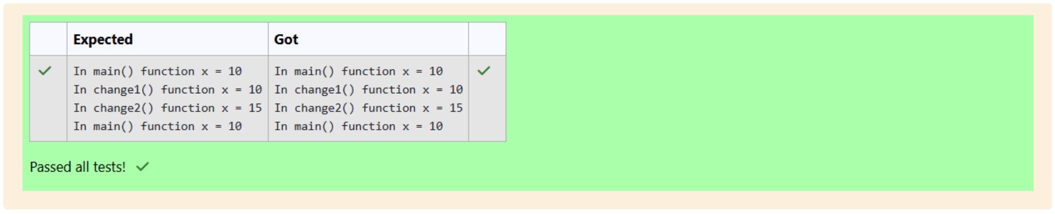
```
#include <stdio.h>

int x = 15;

void change1(int x)
{
   printf("In change1() function x = %d\n", x);
}

void change2()
{
   printf("In change2() function x = %d\n", x);
}

int main()
{
   int x = 10;
   printf("In main() function x = %d\n", x);
   change1(x);
   change2();
```

```
    printf("In main() function x = %d\n", x);
    return 0;
}
```



Answer: (penalty regime: 0 %)

```
#include<stdio.h>
int x=15;
void change1(int x)
{
    printf("In change1() function x = %d\n",x);
}
void change2(){
    printf("In change2() function x = %d\n",x);
}
int main(){
    int x=10;
    printf("In main() function x = %d\n",x);
    change1(x);
    change2();
    printf("In main() function x = %d\n",x);
    return 0;
}
```



	Expected	Got	
✓	In main() function x = 10 In change1() function x = 10 In change2() function x = 15 In main() function x = 10	In main() function x = 10 In change1() function x = 10 In change2() function x = 15 In main() function x = 10	✓

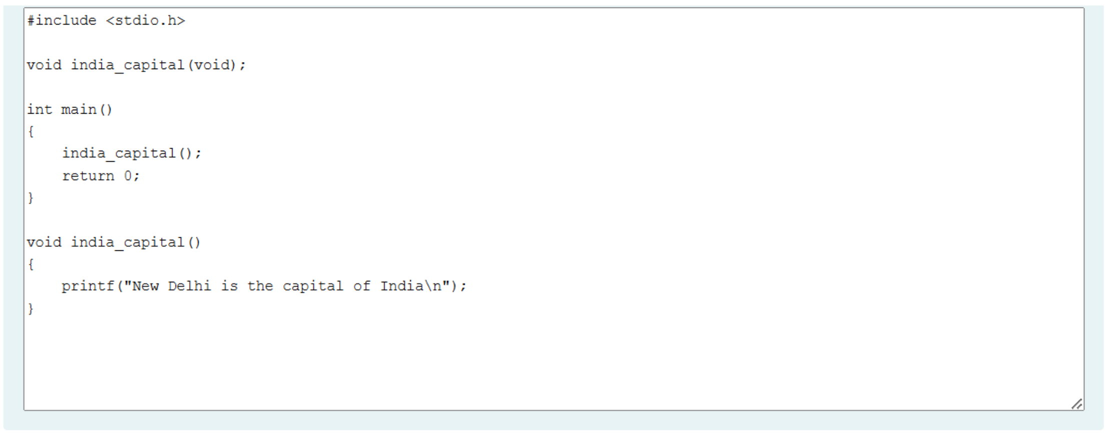
Passed all tests! ✓

9. Let us consider an example of a function without arguments and without return value:

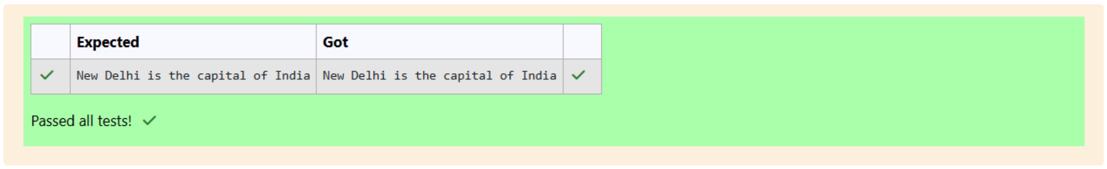
```
#include <stdio.h>

void india_capital(void);

int main()
{

        india_capital();
    return 0;

}

void india_capital()
{

        printf("New Delhi is the capital of India\n");

}
```

In the above sample code the function void india_capital(void); specifies that the function does not receive any arguments and does not return any value to the main() function.

Identify the below errors and correct them.



```
#include <stdio.h>

void india_capital(void);

int main()
{
    india_capital();
    return 0;
}

void india_capital()
{
    printf("New Delhi is the capital of India\n");
}
```



	Expected	Got	
✓	New Delhi is the capital of India	New Delhi is the capital of India	✓

Passed all tests! ✓

10. Write a **C** program to demonstrate functions without arguments and without return value.

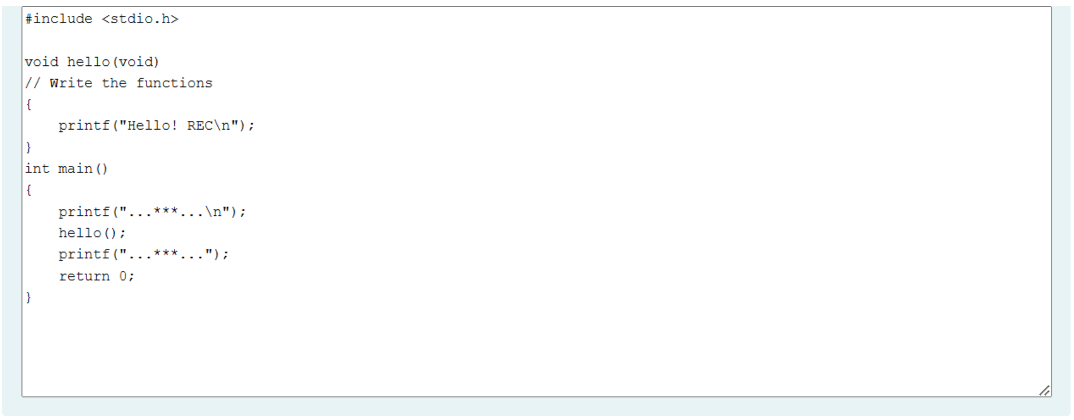
Write the functions **print()** and **hello()**.

The output is:

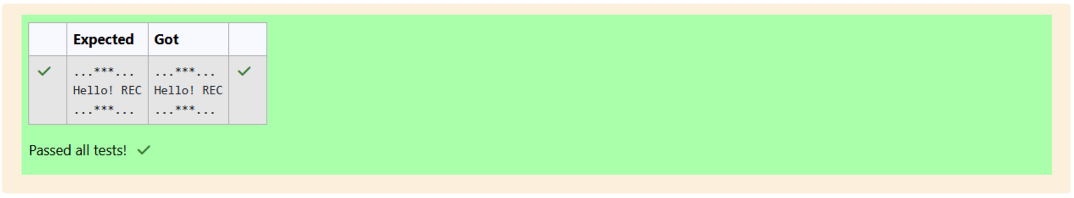
...***...

Hello! REC

...***...



```
#include <stdio.h>

void hello(void)
// Write the functions
{
    printf("Hello! REC\n");
}
int main()
{
    printf("...***...\n");
    hello();
    printf("...***...");
    return 0;
}
```



	Expected	Got	
✓	...***... Hello! REC ...***...	...***... Hello! REC ...***...	✓

Passed all tests! ✓

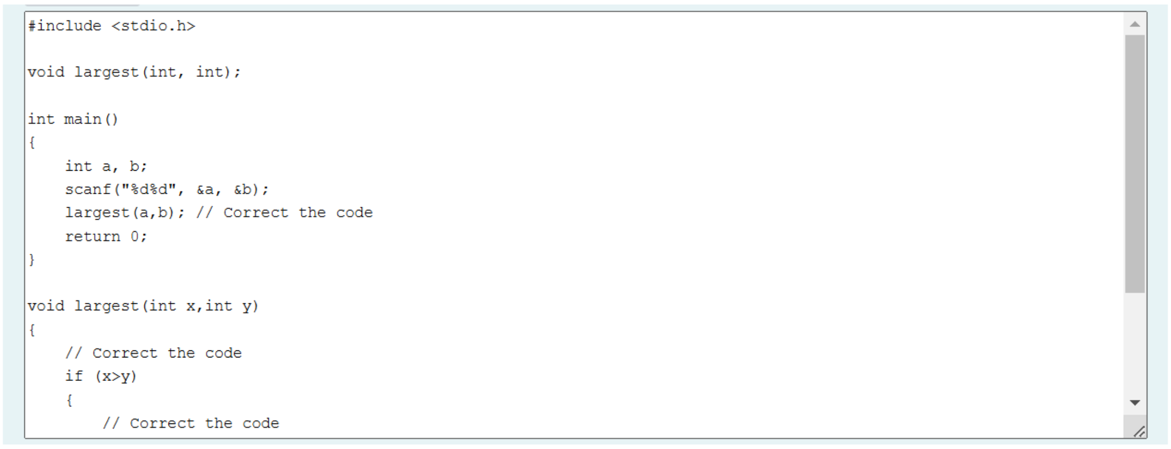
11. Let us consider an example of a function with arguments and without return value:

```
#include <stdio.h>

void largest(int, int);

int main()
{

        int a, b;

        printf("Enter two numbers : ");

        scanf("%d%d" , &a, &b);

        largest(a, b);
    return 0;

}

void largest(int x, int y)
{

        if (x > y)
    {

                printf("Largest element = %d\n", x);

        }
    else
    {
```

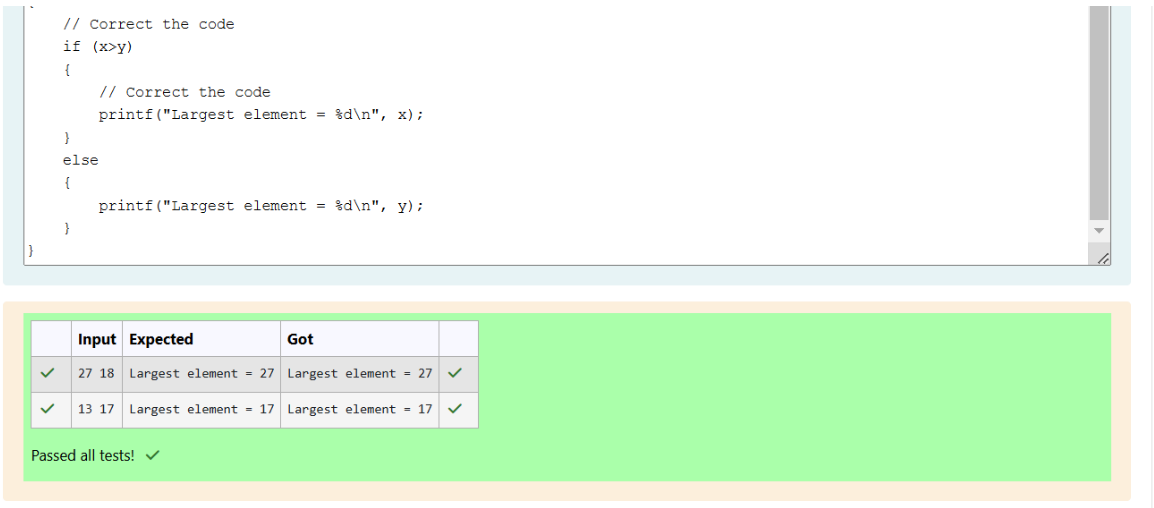
printf("Largest element = %d\n", y);

}

} Fill in the missing code in the below program to find the largest of two numbers using **largest()** function.



```
#include <stdio.h>

void largest(int, int);

int main()
{
    int a, b;
    scanf("%d%d", &a, &b);
    largest(a,b); // Correct the code
    return 0;
}

void largest(int x,int y)
{
    // Correct the code
    if (x>y)
    {
        // Correct the code
```



```
    // Correct the code
    if (x>y)
    {
        // Correct the code
        printf("Largest element = %d\n", x);
    }
    else
    {
        printf("Largest element = %d\n", y);
    }
}
```

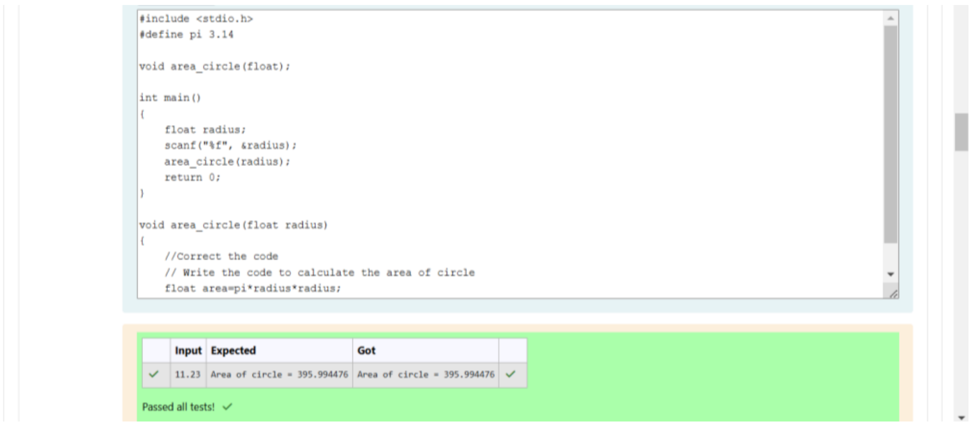
	Input	Expected	Got	
✓	27 18	Largest element = 27	Largest element = 27	✓
✓	13 17	Largest element = 17	Largest element = 17	✓

Passed all tests! ✓

12. Fill the missing code to understand the concept of a function with arguments and without return value.

Note: Take **pi** value as **3.14**

The below code is to find the area of circle using functions.



```c
#include <stdio.h>
#define pi 3.14

void area_circle(float);

int main()
{
    float radius;
    scanf("%f", &radius);
    area_circle(radius);
    return 0;
}

void area_circle(float radius)
{
    //Correct the code
    // Write the code to calculate the area of circle
    float area=pi*radius*radius;
```

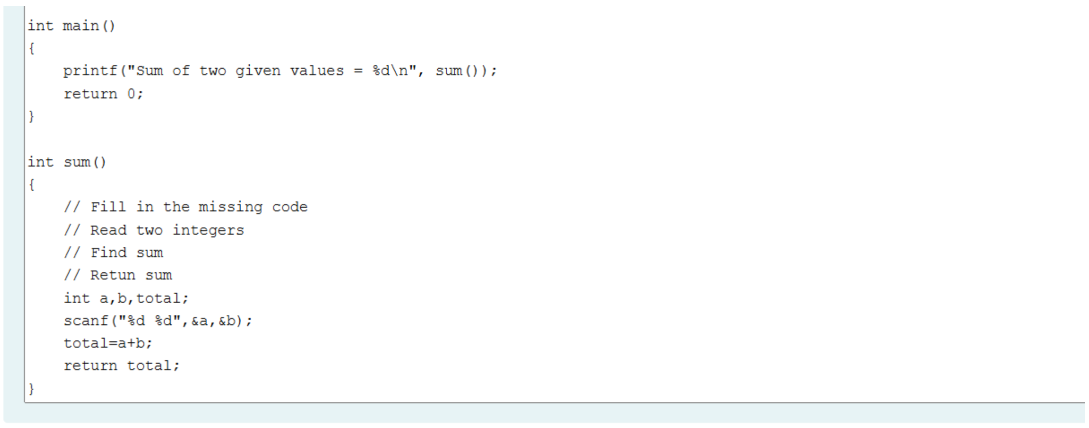
	Input	Expected	Got	
✓	11.23	Area of circle = 395.994476	Area of circle = 395.994476	✓

Passed all tests! ✓

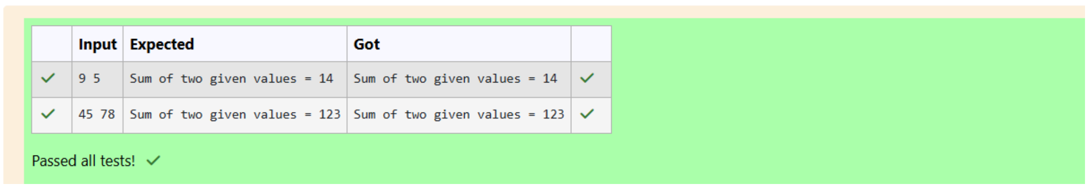
13. #include <stdio.h>

```c
int sum(void);

int main()
{

        printf("\nSum of two given values = %d\n", sum());
    return 0;

}

int sum() {

        int a, b, total;

        printf("Enter two numbers : ");

        scanf("%d%d", &a, &b);

        total = a + b;

        return total;

}
```

Fill in the missing code in the below program to find sum of two integers.



```
int main()
{
    printf("Sum of two given values = %d\n", sum());
    return 0;
}

int sum()
{
    // Fill in the missing code
    // Read two integers
    // Find sum
    // Retun sum
    int a,b,total;
    scanf("%d %d",&a,&b);
    total=a+b;
    return total;
}
```



	Input	Expected	Got	
✓	9 5	Sum of two given values = 14	Sum of two given values = 14	✓
✓	45 78	Sum of two given values = 123	Sum of two given values = 123	✓

Passed all tests! ✓

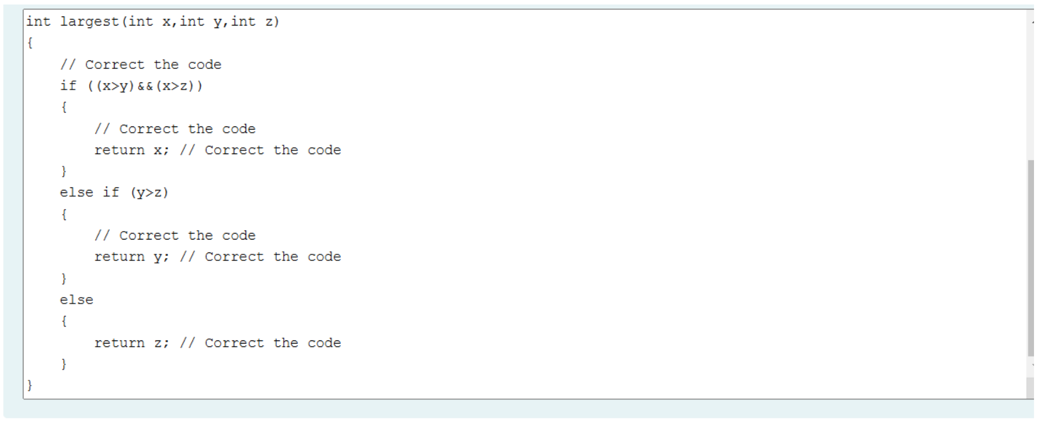
14. #include <stdio.h>

```
int largest(int, int, int);

int main()
{

        int a, b, c;

        printf("Enter three numbers : ");

        scanf("%d%d%d" , &a, &b, &c);

        printf(" Largest of the given three numbers = %d\n", largest(a, b, c));
    return 0;

}

int largest(int x, int y, int z)
{

        if ((x > y) && (x > z))
    {

                return x;

        }
    else if (y > z)
    {
```

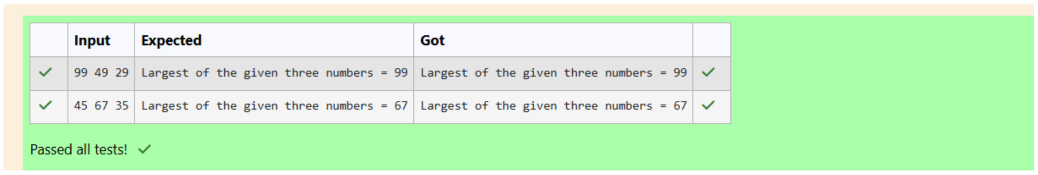
```
                return y;
        }
      else
      {
                return z;
        }
}
```

In the above sample code the function int largest(int, int, int); specifies that the function receives three values and returns a value to the **calling function**.

Fill in the missing code in the below program to find the largest of three numbers using **largest()** function.



```
int largest(int x,int y,int z)
{
    // Correct the code
    if ((x>y)&&(x>z))
    {
        // Correct the code
        return x; // Correct the code
    }
    else if (y>z)
    {
        // Correct the code
        return y; // Correct the code
    }
    else
    {
        return z; // Correct the code
    }
}
```

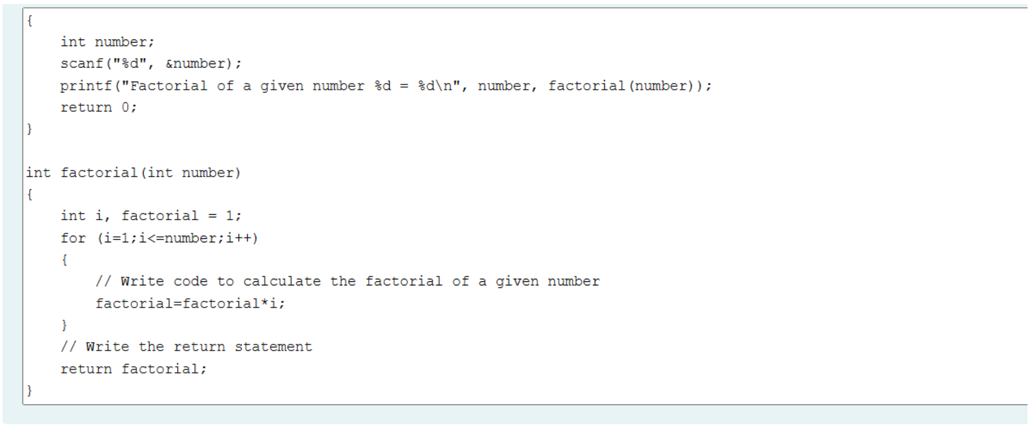


	Input	Expected	Got	
✓	99 49 29	Largest of the given three numbers = 99	Largest of the given three numbers = 99	✓
✓	45 67 35	Largest of the given three numbers = 67	Largest of the given three numbers = 67	✓

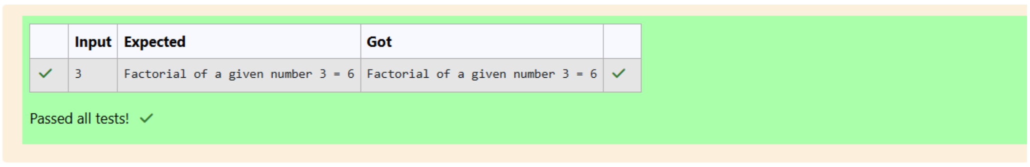
Passed all tests! ✓

15. Fill in the missing code in the below code to understand about function with arguments and with return value.

The below code is to find the factorial of a given number using functions.



```
{
    int number;
    scanf("%d", &number);
    printf("Factorial of a given number %d = %d\n", number, factorial(number));
    return 0;
}

int factorial(int number)
{
    int i, factorial = 1;
    for (i=1;i<=number;i++)
    {
        // Write code to calculate the factorial of a given number
        factorial=factorial*i;
    }
    // Write the return statement
    return factorial;
}
```



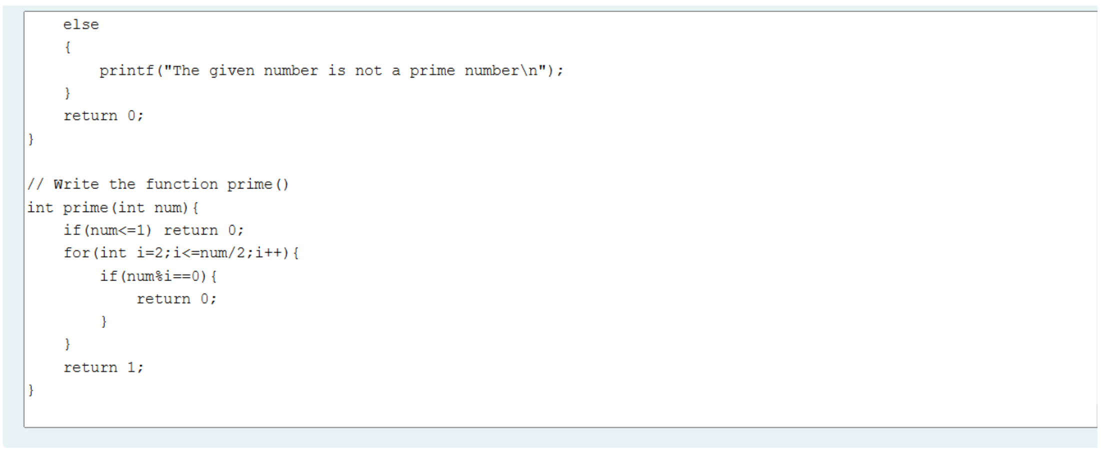
	Input	Expected	Got	
✓	3	Factorial of a given number 3 = 6	Factorial of a given number 3 = 6	✓

Passed all tests! ✓

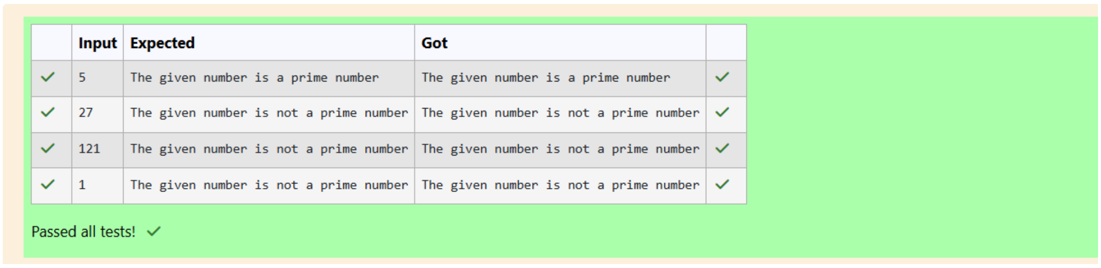
16. Write a **C** program to demonstrate functions without arguments and with return value.

The below code is used to check whether the given number is a prime number or not.

Write the function **prime()**.



```
    else
    {
        printf("The given number is not a prime number\n");
    }
    return 0;
}

// Write the function prime()
int prime(int num){
    if(num<=1) return 0;
    for(int i=2;i<=num/2;i++){
        if(num%i==0){
            return 0;
        }
    }
    return 1;
}
```



	Input	Expected	Got	
✓	5	The given number is a prime number	The given number is a prime number	✓
✓	27	The given number is not a prime number	The given number is not a prime number	✓
✓	121	The given number is not a prime number	The given number is not a prime number	✓
✓	1	The given number is not a prime number	The given number is not a prime number	✓

Passed all tests! ✓