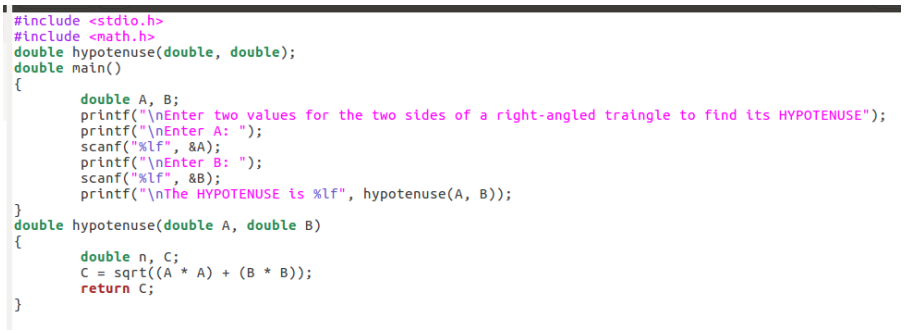
LPRACTICE LAB ASSIGNMENT 6

1. Define a function called hypotenuse that calculates the length of the hypotenuse of a right triangle when the other two sides are given. The function should take two arguments of type double and return the hypotenuse as a double.

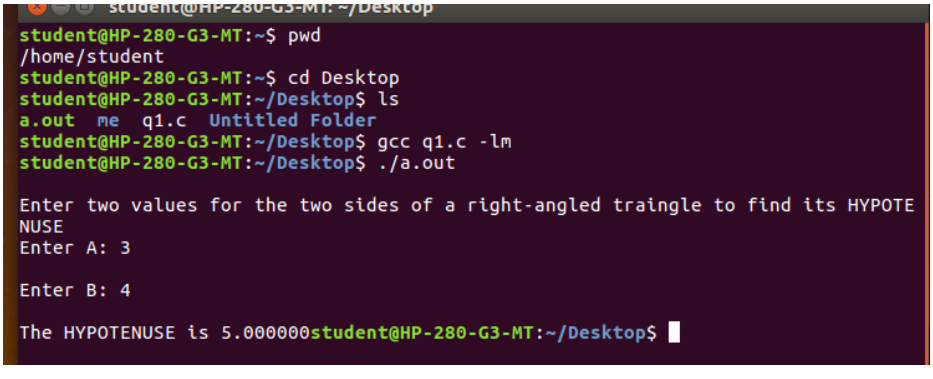
CODE

```
#include <stdio.h>

#include <math.h>

double hypotenuse(double, double);

double main()

{

double A, B;

printf("\nEnter two values for the two sides of a right-angled traingle to find its HYPOTENUSE");

printf("\nEnter A: ");

scanf("%lf", &A);

printf("\nEnter B: ");

scanf("%lf", &B);

printf("\nThe HYPOTENUSE is %lf", hypotenuse(A, B));

}

double hypotenuse(double A, double B)

{

double n, C;

C = sqrt((A * A) + (B * B));

return C;

}
```



```
#include <stdio.h>
#include <math.h>
double hypotenuse(double, double);
double main()
{
        double A, B;
        printf("\nEnter two values for the two sides of a right-angled traingle to find its HYPOTENUSE");
        printf("\nEnter A: ");
        scanf("%lf", &A);
        printf("\nEnter B: ");
        scanf("%lf", &B);
        printf("\nThe HYPOTENUSE is %lf", hypotenuse(A, B));
}
double hypotenuse(double A, double B)
{
        double n, C;
        C = sqrt((A * A) + (B * B));
        return C;
}
```

SS of the OUTPUT



```
student@HP-280-G3-MT:~$ pwd
/home/student
student@HP-280-G3-MT:~$ cd Desktop
student@HP-280-G3-MT:~/Desktop$ ls
a.out  me  q1.c  Untitled Folder
student@HP-280-G3-MT:~/Desktop$ gcc q1.c -lm
student@HP-280-G3-MT:~/Desktop$ ./a.out

Enter two values for the two sides of a right-angled traingle to find its HYPOTE
NUSE
Enter A: 3

Enter B: 4

The HYPOTENUSE is 5.000000student@HP-280-G3-MT:~/Desktop$
```

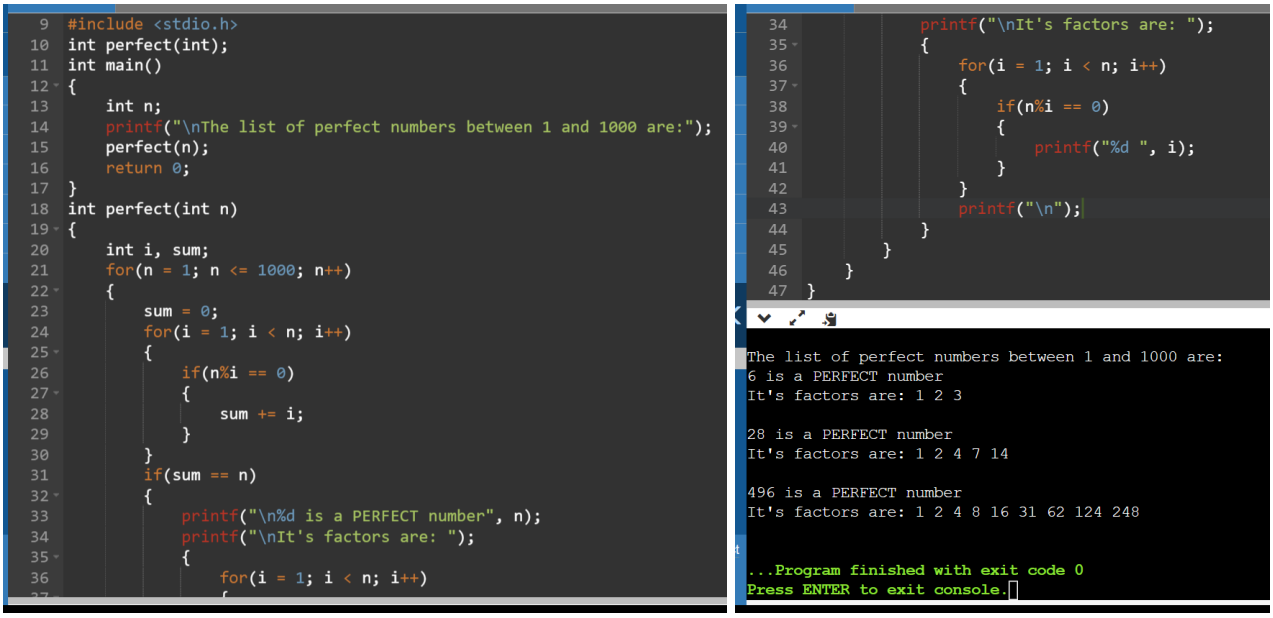
2. An integer number is said to be a perfect number if its factors, including1 (but not the number itself), sum to the number. For example, 6 is a perfect number because 6 =1 + 2 + 3. Write a function perfect( )that determines if parameter number is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect.

CODE

```
#include <stdio.h>
int perfect(int);
int main()
{
   int n;
   printf("\nThe list of perfect numbers between 1 and 1000 are:");
   perfect(n);
   return 0;
}
int perfect(int n)
{
   int i, sum;
   for(n = 1; n <= 1000; n++)
   {
      sum = 0;
      for(i = 1; i < n; i++)
      {
         if(n%i == 0)
         {
            sum += i;
         }
      }
      if(sum == n)
```

```
        {
            printf("\n%d is a PERFECT number", n);
            printf("\nIt's factors are: ");
            {
                for(i = 1; i < n; i++)
                {
                    if(n%i == 0)
                    {
                        printf("%d ", i);
                    }
                }
                printf("\n");
            }
        }
    }
}
```

SS of the OUTPUT



```
#include <stdio.h>
int perfect(int);
int main()
{
    int n;
    printf("\nThe list of perfect numbers between 1 and 1000 are:");
    perfect(n);
    return 0;
}
int perfect(int n)
{
    int i, sum;
    for(n = 1; n <= 1000; n++)
    {
        sum = 0;
        for(i = 1; i < n; i++)
        {
            if(n%i == 0)
            {
                sum += i;
            }
        }
        if(sum == n)
        {
            printf("\n%d is a PERFECT number", n);
            printf("\nIt's factors are: ");
            {
                for(i = 1; i < n; i++)
```

```
            printf("\nIt's factors are: ");
            {
                for(i = 1; i < n; i++)
                {
                    if(n%i == 0)
                    {
                        printf("%d ", i);
                    }
                }
                printf("\n");
            }
        }
    }
}
```

```
The list of perfect numbers between 1 and 1000 are:
6 is a PERFECT number
It's factors are: 1 2 3

28 is a PERFECT number
It's factors are: 1 2 4 7 14

496 is a PERFECT number
It's factors are: 1 2 4 8 16 31 62 124 248


...Program finished with exit code 0
Press ENTER to exit console.
```

3. Write a program using function that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the function should return 1367.

CODE

```
#include <stdio.h>
int reverse(int);
int main()
{
   int n;
   printf("\nEnter any integer to find out its reverse: ");
   scanf("%d", &n);
   printf("The reverse of the entered integer value is %d", reverse(n));
}
int reverse(int n)
{
   int rev = 0, r;
   if(n == 0)
   return 0;
   while(n != 0)
   {
      r = n%10;
      rev = ((rev * 10) + r);
      n /= 10;
   }
   return rev;
}
```

SS of the OUTPUT

```
Enter any integer to find out its reverse: 0
The reverse of the entered integer value is 0

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter any integer to find out its reverse: 1367
The reverse of the entered integer value is 7631

...Program finished with exit code 0
Press ENTER to exit console.
```

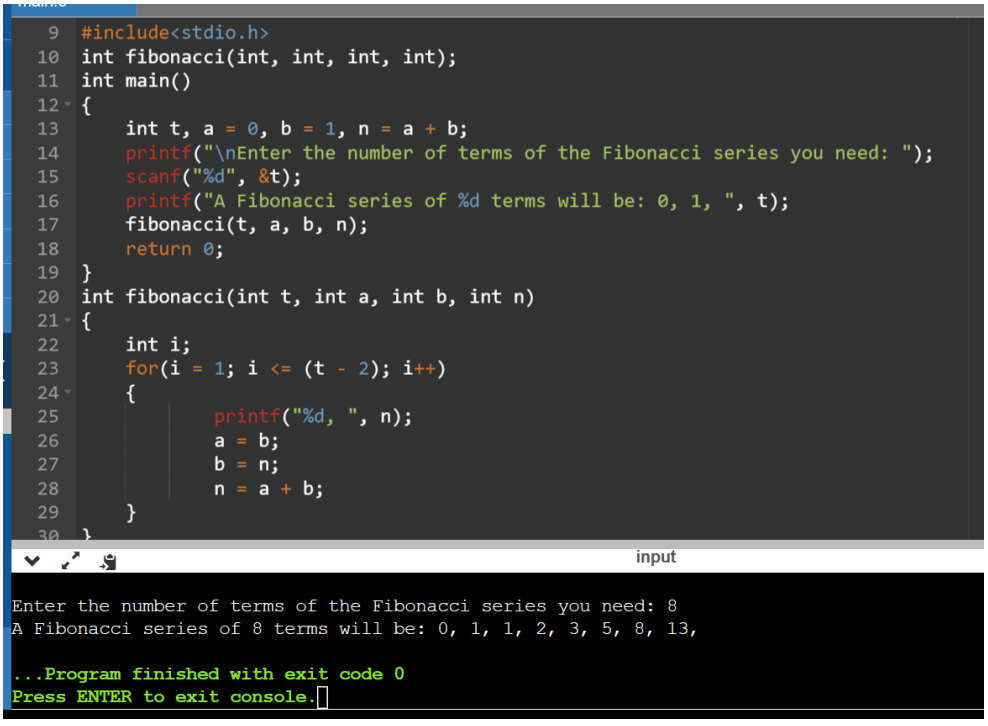
```
Enter any integer to find out its reverse: -3671
The reverse of the entered integer value is -1763

...Program finished with exit code 0
Press ENTER to exit console.
```

4. Write a Program to print the Fibonacci Series with the help of functions. A Fibonacci Series is a series of numbers where the next number in the series is equal to the sum of previous 2 numbers. Example:-A Fibonacci series of 8 terms will be -0, 1, 1, 2, 3, 5, 8, 13.

CODE

```
#include<stdio.h>
int fibonacci(int, int, int, int);
int main()
{
   int t, a = 0, b = 1, n = a + b;
   printf("\nEnter the number of terms of the Fibonacci series you need: ");
   scanf("%d", &t);
   printf("A Fibonacci series of %d terms will be: 0, 1, ", t);
   fibonacci(t, a, b, n);
   return 0;
}
int fibonacci(int t, int a, int b, int n)
{
   int i;
   for(i = 1; i <= (t - 2); i++)
   {
        printf("%d, ", n);
        a = b;
        b = n;
        n = a + b;
   }
}
```

SS of the OUTPUT



5. Write a function multiple that determines for a pair of integers whether the second integer is a multiple of the first. The function should take two integer arguments and return 1(true) if the second is a multiple of the first and 0 (false) otherwise.

CODE

```
#include<stdio.h>
```

```
int multiple(int, int);
int main()
{
   int n, m;
   printf("\nEnter any integer: ");
   scanf("%d", &n);
   printf("\nEnter an integer to check if it's a multiple of the previous number: ");
   scanf("%d", &m);
   printf("%d", multiple(n, m));
   return 0;
}
int multiple(int n, int m)
{
   if(m%n == 0)
   return 1;
   else
   return 0;
}
```

```
#include<stdio.h>
int multiple(int, int);
int main()
{
    int n, m;
    printf("\nEnter any integer: ");
    scanf("%d", &n);
    printf("\nEnter an integer to check if it's a multiple of the previous number: ");
    scanf("%d", &m);
    printf("%d", multiple(n, m));
    return 0;
}
int multiple(int n, int m)
{
    if(m%n == 0)
    return 1;
    else
    return 0;
}
```

input

```
Enter any integer: 7

Enter an integer to check if it's a multiple of the previous number: 48
0

...Program finished with exit code 0
Press ENTER to exit console.
```

SS of the OUTPUT

```
Enter any integer: 7

Enter an integer to check if it's a multiple of the previous number: 49
1

...Program finished with exit code 0
Press ENTER to exit console.
```