Green University of Bangladesh
Department of Computer Science and Engineering

Program: CSE (Regular) **I** Spring 2023

Lab Report 4

Course Title: Computational Thinking and Problem Solving
Course Code: CSE 100
Section: 231 D1

Student Details

Name	ID No.
Mahmuda Akter Nadia	231002001

Lab Date: 30/05/23
Submission Date: 15/06/23
Course Instructor: Saurav Chandra Das

Lab Report Evaluation	
Marks:	Signature:
Comment:	Date:

Lab Report 4.1

Title: Find whether a given number is a prime number or not

Given Algorithm:

```
Step 1: Begin
Step 2: Display "Enter a number: "
Step 3: Read n
Step 4: Initialize c to 0
Step 5: For i = 1 to n, do
Step 5.1: If "n%i==0"
Step 5.1.1: Increment c by 1
Step 5.2: EndIf;
Step 5.3: Increment i by 1
Step 6: EndFor;
Step 7: If "c<=2"
Step 7.1: Display n " is prime number"
Step 8: Else
Step 8.1: Display n " is not prime number"
Step 9: EndIf;
Step 10: End.
```

1. Introduction:

The objective of this lab is to develop a C program that determines whether a given number is a prime number or not followed by the mentioned algorithm. The program should prompt the user to enter a number, check its divisibility by iterating from 1 to the number, and display the result indicating whether it is a prime number or not.

2. Background:

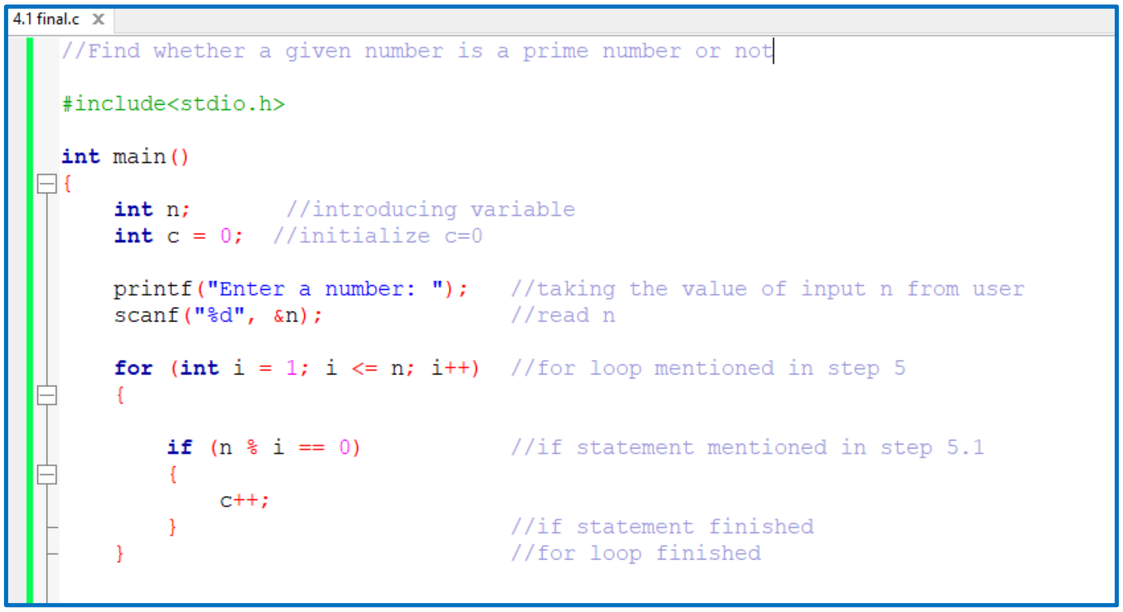
A prime number is a positive integer greater than 1 that has no divisors other than 1 and itself. This program can be achieved using loop and conditional statements in C.

3. Design:

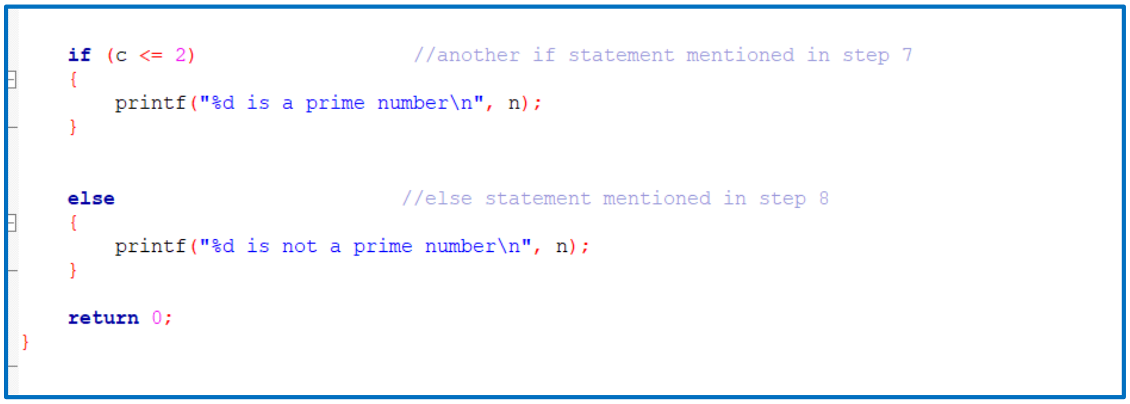
To find the prime number in C, the given algorithm is used to check any number is prime or not:

```
Step 1: Begin
Step 2: Display "Enter a number: "
Step 3: Read n
Step 4: Initialize c to 0

Step 5: For i = 1 to n, do
Step 5.1: If "n%i==0"
Step 5.1.1: Increment c by 1
Step 5.2: EndIf;
Step 5.3: Increment i by 1

Step 6: EndFor;

Step 7: If "c<=2"
Step 7.1: Display n " is prime number"

Step 8: Else
Step 8.1: Display n " is not prime number"
Step 9: EndIf;
Step 10: End.
```

4. Implementation:



```c
//Find whether a given number is a prime number or not

#include<stdio.h>

int main()
{
    int n;       //introducing variable
    int c = 0;  //initialize c=0

    printf("Enter a number: ");   //taking the value of input n from user
    scanf("%d", &n);              //read n

    for (int i = 1; i <= n; i++)  //for loop mentioned in step 5
    {

        if (n % i == 0)           //if statement mentioned in step 5.1
        {
            c++;
        }                         //if statement finished
    }                             //for loop finished
```



```c
    if (c <= 2)                  //another if statement mentioned in step 7
    {
        printf("%d is a prime number\n", n);
    }

    else                        //else statement mentioned in step 8
    {
        printf("%d is not a prime number\n", n);
    }

    return 0;
}
```

5. Experimental Setup:

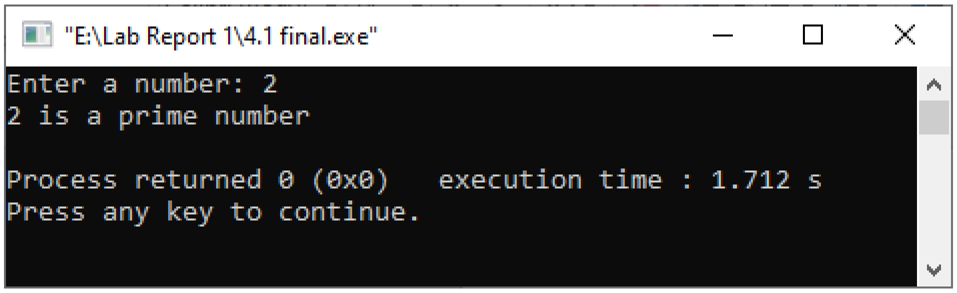
The program was implemented and tested using an Integrated Development Environment named “CodeBlocks” on a Microsoft operating system. The input for each test case was manually provided during the program execution.

6. Results and Discussion:

To ensure the correctness of the program, it has been conducted several test cases. Here are the results some:

Test Case 1:

- **Input:** 2
- **Output:** 2 is a prime number



"E:\Lab Report 1\4.1 final.exe"

```
Enter a number: 2
2 is a prime number

Process returned 0 (0x0)   execution time : 1.712 s
Press any key to continue.
```

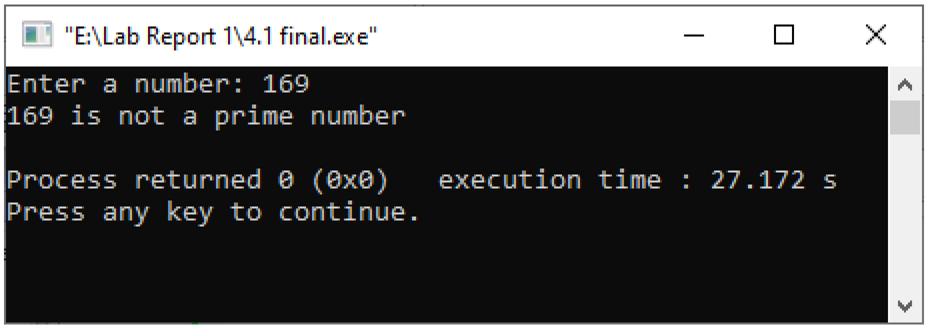
- **Explanation:**

  In this case, **2** is only divided by 1 and itself without any remainder. So, the program successfully displays 2 as a prime number.

Test Case 2:

- **Input:** 169

- **Output:** 169 is not a prime number

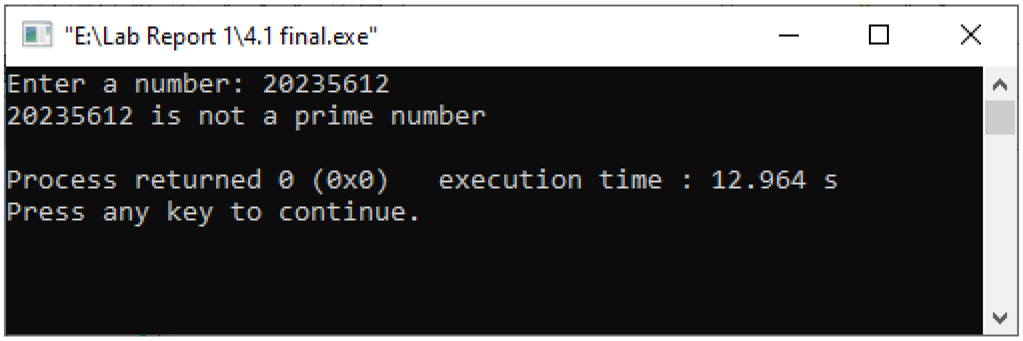

```
"E:\Lab Report 1\4.1 final.exe"
Enter a number: 169
169 is not a prime number

Process returned 0 (0x0)   execution time : 27.172 s
Press any key to continue.
```

- **Explanation:**

  In this case, **169** is product of 13 including 1 and itself 169. So, **169** is not a prime number and the program correctly identifies it.

Test Case 3:

- **Input:** 20235612
- **Output:** 20235612 is not a prime number


```
"E:\Lab Report 1\4.1 final.exe"
Enter a number: 20235612
20235612 is not a prime number

Process returned 0 (0x0)   execution time : 12.964 s
Press any key to continue.
```

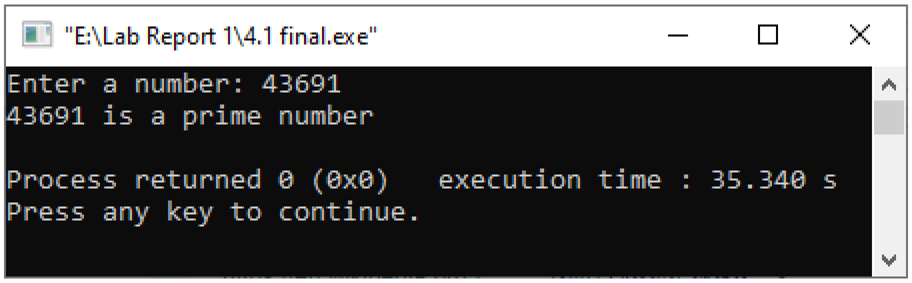
- **Explanation:**

  In this case, **20235612** is also not a prime number since 20235612 =2*10117806. So, the program correctly displays that the given number is not prime.

Test Case 4:

- **Input:** 43691

- **Output:** 43691 is not a prime number



"E:\Lab Report 1\4.1 final.exe"

```
Enter a number: 43691
43691 is a prime number

Process returned 0 (0x0)   execution time : 35.340 s
Press any key to continue.
```

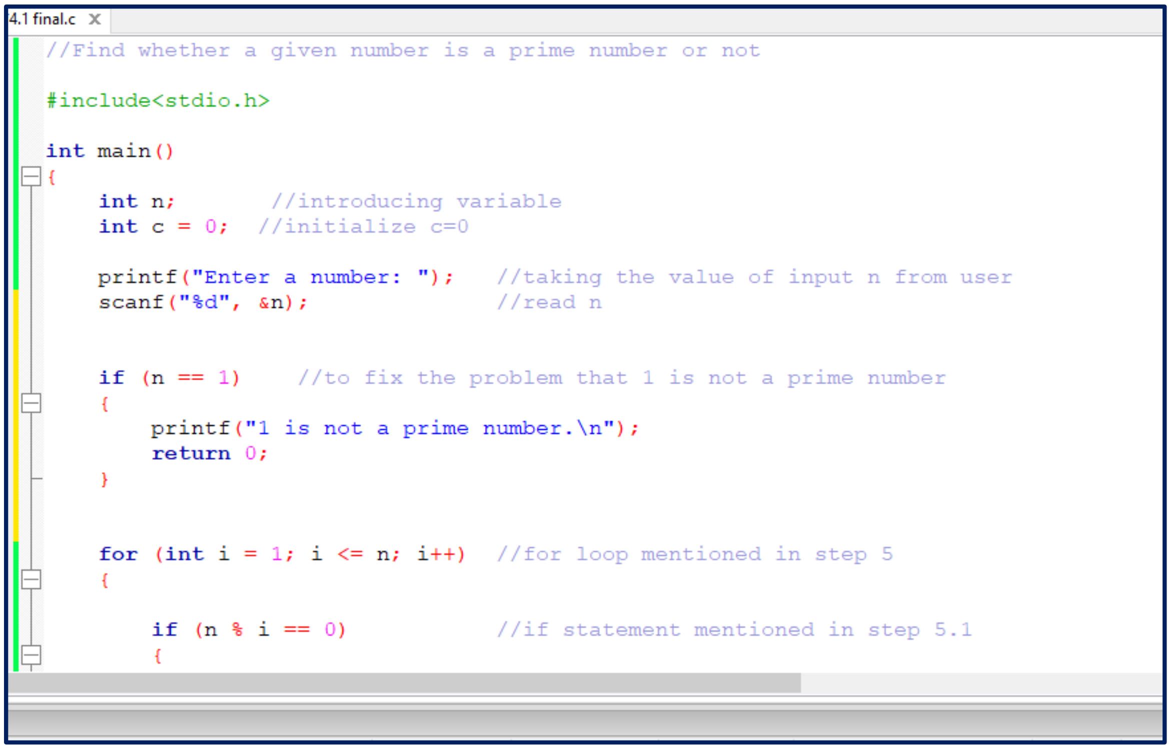
- **Explanation:**

  In this case, **43691** has no divisors other than 1 and itself 43691. SO, it displays the output: 43691 is not a prime number.

For any value of the double numbers, this program can evaluate the maximum between three numbers.

7. Analysis and Discussions:

The program achieves its objective to determine the number prime or not by checking its divisibility and counting the number of divisors. But the program shows 1 as a prime number. To fix the problem, we can add a condition before for loop to check if the input number 'n' is equal to 1 separately. Here is the updated version of the code before the loop and the rest code will be the same so here it is not added again:



```c
//Find whether a given number is a prime number or not

#include<stdio.h>

int main()
{
    int n;        //introducing variable
    int c = 0;   //initialize c=0

    printf("Enter a number: ");    //taking the value of input n from user
    scanf("%d", &n);               //read n


    if (n == 1)     //to fix the problem that 1 is not a prime number
    {
        printf("1 is not a prime number.\n");
        return 0;
    }


    for (int i = 1; i <= n; i++)   //for loop mentioned in step 5
    {

        if (n % i == 0)            //if statement mentioned in step 5.1
        {
```

This program helps us to understand the important concept of programming such as input validation, variable declaration, loop structure, counting divisors and conditional statements.

8. Summary:

Implementing, this lab exercise helped strengthen our understanding of important concepts of programming like problem-solving skills, algorithm thinking, validation and testing, while also providing hands-on experience in C programming.

Lab Report 4.2

Title: Display Fibonacci Sequence

1. Introduction:

The objective of this lab is to develop a C program that generates and prints the Fibonacci series until a given number. The program should prompt the user to enter a number, calculate the Fibonacci series up to that number, and display the series.

2. Background:

In mathematics, the Fibonacci series is a sequence of numbers in which each number is the sum of the two preceding ones, usually starting with 0 and 1, that is defined by the recurrence relation. This lab aims to create a C program that generates and prints the Fibonacci series up to a given number. The user will input the maximum number up to which the series should be printed, and the program will calculate and display the Fibonacci numbers that are less than the given number.

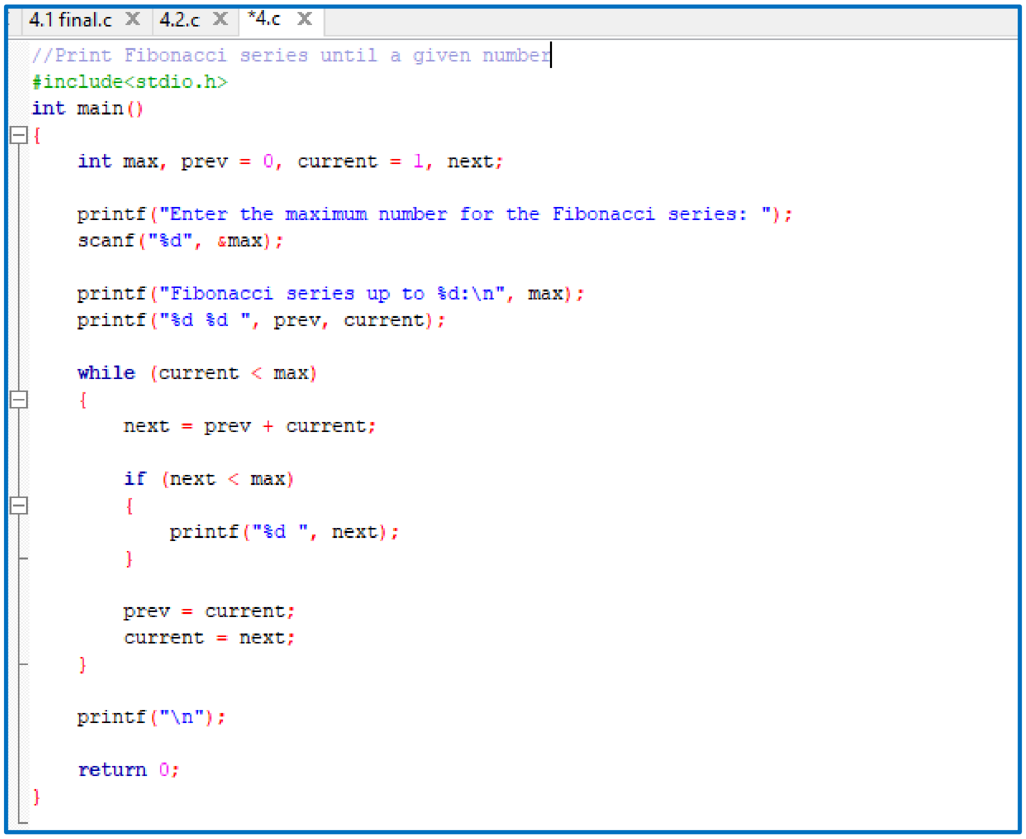
3. Design:

To display the Fibonacci sequence in C, the mentioned steps are followed:

1. Start the program
2. Prompt the user to enter a number as the maximum value for the Fibonacci series
3. Read and store the input in the variable 'max'
4. Declare and initialize variables 'prev' and 'current' as 0 and 1 respectively
5. Display the first two Fibonacci numbers, 0 and 1
6. While 'current' is less than 'max', do steps 7 and 8
7. Calculate the next Fibonacci number, 'next', as the sum of 'prev' and 'current'

8. Display 'next' if it is less than 'max' and update 'prev' and 'current'
9. End the loop
10. End the program

4. Implementation:



4.1 final.c 4.2.c *4.c

```c
//Print Fibonacci series until a given number
#include<stdio.h>
int main()
{
    int max, prev = 0, current = 1, next;

    printf("Enter the maximum number for the Fibonacci series: ");
    scanf("%d", &max);

    printf("Fibonacci series up to %d:\n", max);
    printf("%d %d ", prev, current);

    while (current < max)
    {
        next = prev + current;

        if (next < max)
        {
            printf("%d ", next);
        }

        prev = current;
        current = next;
    }

    printf("\n");

    return 0;
}
```

5. Experimental Setup:

The program was implemented and tested using an Integrated Development Environment named “CodeBlocks” on a Microsoft operating system. The input for each test case was manually provided during the program execution.

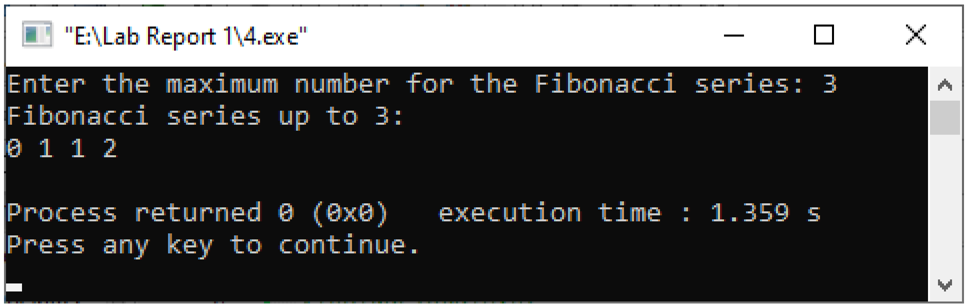
6. Results and Discussion:

The implemented program was tested with various inputs to verify its correctness and accuracy. The following results were obtained:

Test Case 1:

- **Input:** 3
- **Output:**

  Fibonacci series up to 3:
  0 1 1 2



```
Enter the maximum number for the Fibonacci series: 3
Fibonacci series up to 3:
0 1 1 2

Process returned 0 (0x0)   execution time : 1.359 s
Press any key to continue.
```

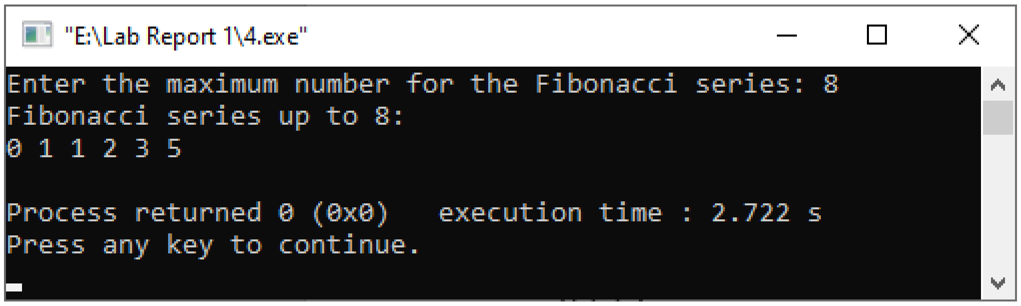
- **Explanation:**

  In this case, a user wants to print Fibonacci series until 3. It finishes on 2 because the next one is 1+2=3, which is equal to 3 that is against the objective of the program. So the program prints all the Fibonacci number below 3.

Test Case 2:

- **Input:** 8
- **Output:**

Fibonacci series up to 3:
0 1 1 2 3 5



```
Enter the maximum number for the Fibonacci series: 8
Fibonacci series up to 8:
0 1 1 2 3 5

Process returned 0 (0x0)   execution time : 2.722 s
Press any key to continue.
```

- **Explanation:**

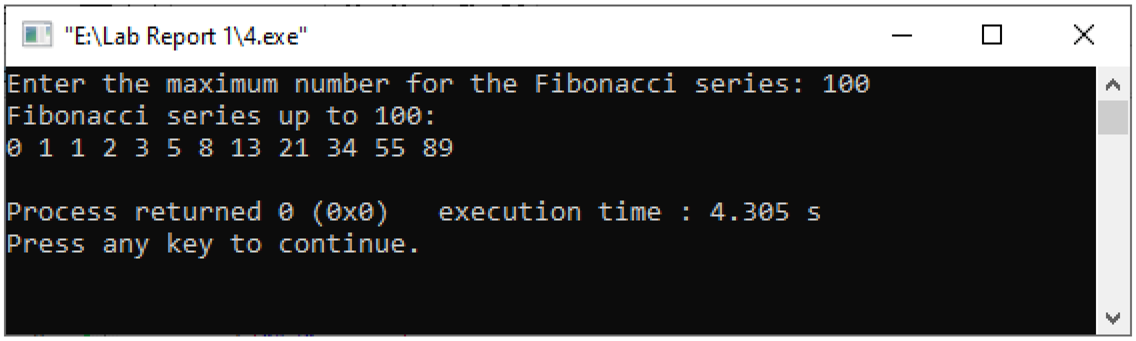
  In this case, a user wants to print Fibonacci series until 8, so the program prints all the Fibonacci number below 8.

Test Case 3:

- **Input:** 100

- **Output:**

  Fibonacci series up to 3:
  0 1 1 2 3 5 8 13 21 34 55 89



```
Enter the maximum number for the Fibonacci series: 100
Fibonacci series up to 100:
0 1 1 2 3 5 8 13 21 34 55 89

Process returned 0 (0x0)   execution time : 4.305 s
Press any key to continue.
```

- **Explanation:**

  In this case, a user wants to print the Fibonacci series until 100. It finishes on 89 because the next one 55+89 = 144 is bigger than 100, which is against the objective of the program. So the program prints all the Fibonacci number below 144.

The program displayed accurate results for a variety of test cases, proving its correctness and functionality.

7. Analysis and Discussions:

The provided C program generates and prints the Fibonacci series up to a given number. It helps us to understand the logic of the Fibonacci Series Generation using loop, and variable declaration.

8. Summary:

This program demonstrates an understanding of loops, variables, arithmetic operations, and input/output functions in C. Executing the program, users can easily obtain the Fibonacci numbers below a specified maximum value.