



**Green University of Bangladesh**  
**Department of Computer Science and Engineering**

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

**Lab Report 1**

**Course Title:** Computational Thinking and Problem Solving

**Course Code:** CSE 100

**Section:** 231 D1

**Student Details**

<b>Name</b>	<b>ID No.</b>
Mahmuda Akter Nadia	231002001

**Lab Date:** 30/05/23

**Submission Date:** 05/06/23

**Course Instructor:** Saurav Chandra Das

<b>Lab Report Evaluation</b>	
Marks:	Signature:
Comment:	Date:

## **Title: Lab Report 1.1**

### **Find the Maximum between two numbers in C**

#### **1. Introduction:**

The purpose of this lab is to develop a program in C programming language that finds the maximum between two numbers. This program is a fundamental exercise in C programming and serves as a building block for more complex applications. By completing this lab, we aim to enhance our understanding of basic programming concepts, such as input, variables and conditional statements.

#### **2. Background:**

Finding the maximum between two numbers involves comparing the values of the two numbers and determining which one is larger. This can be achieved using conditional statements in C. The algorithm compares the values and then displays the maximum value as the output.

#### **3. Design:**

To find the maximum between two numbers in C, I used the following steps:

- i. Algorithm
- ii. Flowchart

The algorithm and flowchart created using the Lucid app are attached below:

#### Algorithm of a C program to find the maximum between two number

**Step 1:** Start

**Step 2:** Input num1, num2

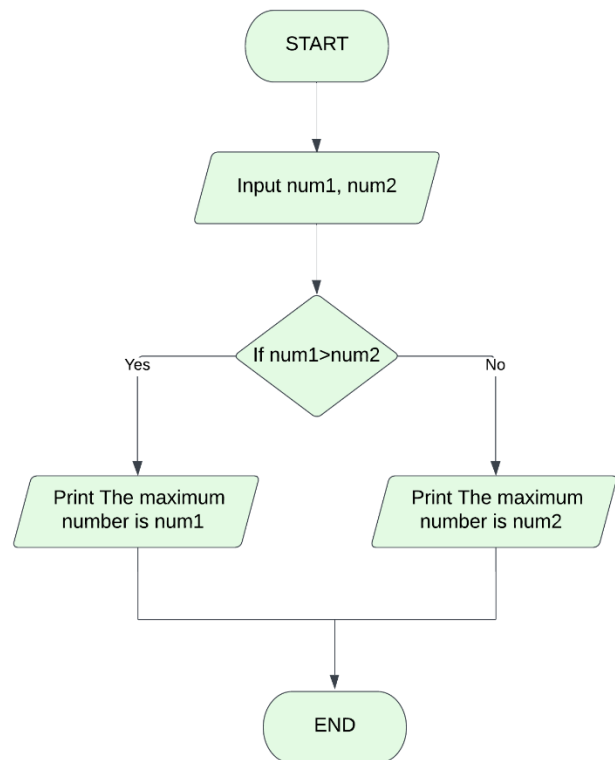
**Step 3:** If num1>num2, go to step 4, otherwise goto step 5

**Step 4:** Print The maximum number is num1, goto step 6

**Step 5:** Print The maximum number is num2, goto step 6

**Step 6:** End

#### Flowchart of a C program to find the maximum between two number



## 4. Implementation:

```
Start here X Lab_Report1 Problem1.c X
1 //Write a C program to find the maximum between two numbers.
2
3 #include<stdio.h>
4
5 int main ()
6
7 {
8     int num1, num2;
9     printf ("Enter two numbers:");
10    scanf ("%d %d", &num1, &num2);
11
12    if (num1>num2)
13    {
14        printf ("The maximum number is %d" , num1);
15    }
16
17    else
18    {
19        printf ("The maximum number is %d" , num2);
20    }
21
22
23    return 0;
24 }
25
```

The program prompts the user to enter two numbers using the scanf function. Then it compares the values of num1 and num2 using an if-else statement. If num1 is greater than num2, the program displays num1 as the maximum number using printf function otherwise it displays the num2.

## 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:

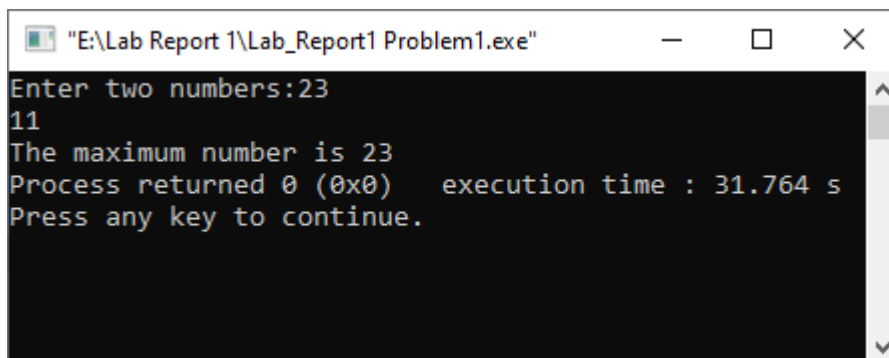
### Test Case 1:

- **Input:**

The first number, **num1** is 23

The second number, **num2** is 11

- **Output:** The maximum number is 23



```
"E:\Lab Report 1\Lab_Report1 Problem1.exe"
Enter two numbers:23
11
The maximum number is 23
Process returned 0 (0x0) execution time : 31.764 s
Press any key to continue.
```

- **Explanation:**

In this case, **num1** is greater than **num2**. So, the program correctly identifies 23 as the maximum number.

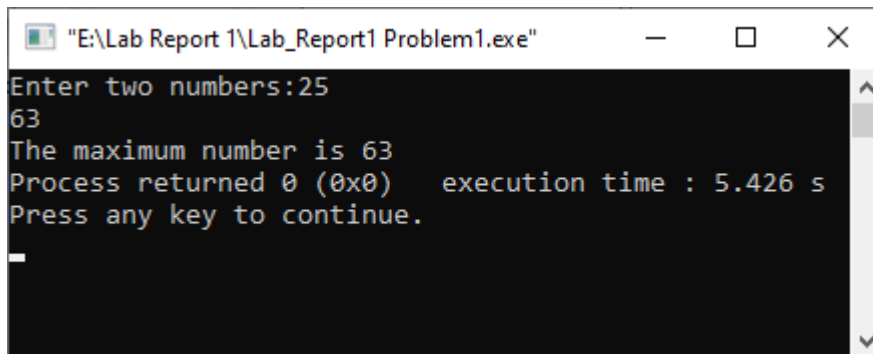
### Test Case 2:

- **Input:**

The first number, **num1** is 25

The second number, **num2** is 63

- **Output:** The maximum number is 63



```
"E:\Lab Report 1\Lab_Report1 Problem1.exe"
Enter two numbers:25
63
The maximum number is 63
Process returned 0 (0x0) execution time : 5.426 s
Press any key to continue.
_
```

- **Explanation:**

In this case, **num2** is greater than **num1**. So, the program correctly displays the value of **num2** as the maximum number.

For any value of the integers, this program can evaluate the maximum between two numbers.

## **7. Analysis and Discussions:**

The program successfully achieves its objective of finding the maximum between two numbers. The algorithm is simple and efficient, making it suitable for basic calculations involving two numbers.

One potential improvement could be to handle scenarios where the two input numbers are equal. Currently, the program considers them separately, resulting in two separate maximum values. By adding an additional condition to check for equality, we can handle such cases and output a meaningful response.

## **8. Summary:**

This lab exercise helped strengthen our understanding of fundamental concepts of programming like how to take integer input, use conditional statements and variables, to print the result finally, while also providing hands-on experience in C programming.

## **Title: Lab Report 1.2**

### **Check whether a number is negative, positive, or zero in C**

#### **1. Introduction:**

The objective of this lab is to create a program in C programming language to determine whether a given number is negative, positive, or zero. The program demonstrates the fundamental concept of decision-making in programming.

#### **2. Background:**

It is needed to check various conditional statements in C to determine the type of given number. The algorithm compares the conditions and then displays the type of number whether to be negative, positive, or zero as the output.

#### **3. Design:**

To determine the type of number in C, the following steps are used:

- i. Algorithm
- ii. Flowchart

Here it shows the total of 8 steps in the Algorithm to write this program.

The algorithm and flowchart created using the Lucid app are attached below:

#### Algorithm of a C program whether a number is negative, positive or zero

**Step 1:** Start

**Step 2:** Input num

**Step 3:** If num>0, go to step 4, otherwise goto step 5

**Step 4:** Print The number num is Positive, goto step 8

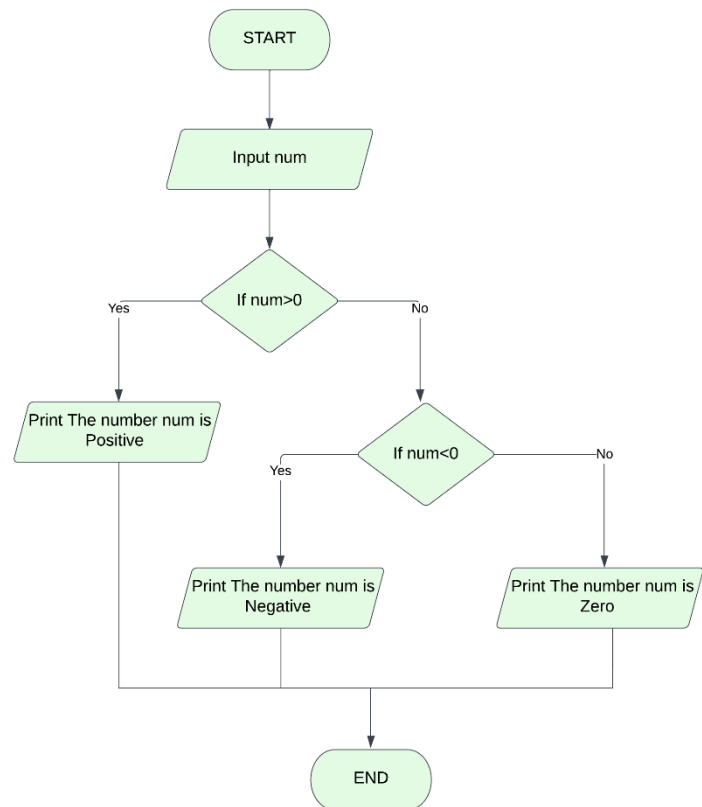
**Step 5:** If num<0, go to step 6, otherwise goto step 7

**Step 6:** Print The number num is Negative, go to step 8, otherwise goto step 7

**Step 7:** Print The number num is Zero, go to step 8

**Step 8:** End

#### Flowchart of a C program whether a number is negative, positive or zero



## 4. Implementation:

```
Lab_Report1 Problem1.c X Lab_Report1 Problem 2.c X
// Write a C program to check whether a number is negative, positive, or zero.

#include<stdio.h>
int main ()
{
    int num;

    printf ("Enter a number:");
    scanf ("%d", &num);

    if (num>0)
    {
        printf ("The number %d is Positive", num);
    }

    else if (num<0)
    {
        printf ("The number %d is Negative", num);
    }

    else
    {
        printf ("The number %d is Zero", num);
    }

    return 0;
}
```



The program prompts the user to enter a number (num) using the scanf function. Then it checks the value of num using conditional statement. If the criteria of certain conditions of if statement / else if statement / else statement meets, the program displays the type of the number using printf function.

## 5. Experimental Setup:

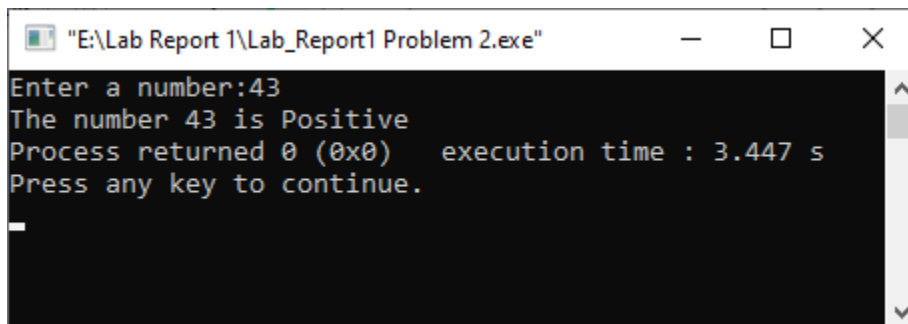
The program was implemented and tested using an Integrated Development Environment named "CodeBlocks" on a Microsoft operating system. The input for each test case was manually provided during the program execution.

## 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:** 43
- **Output:** The number 43 is Positive



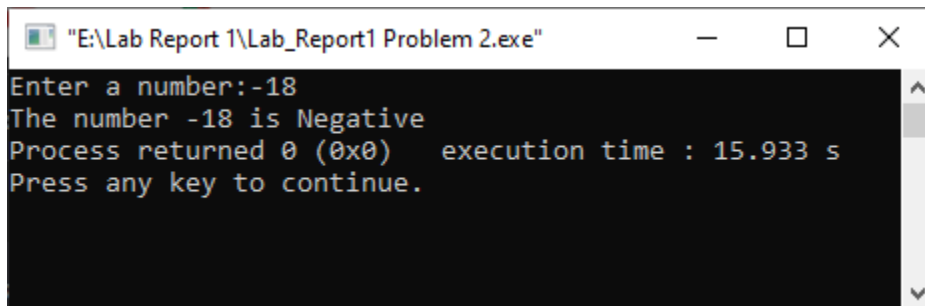
```
"E:\Lab Report 1\Lab_Report1 Problem 2.exe"
Enter a number:43
The number 43 is Positive
Process returned 0 (0x0)   execution time : 3.447 s
Press any key to continue.
```

- **Explanation:**

In this case, **num** is greater than **0**. So, the program correctly identifies 43 as the Positive number.

**Test Case 2:**

- **Input:** -18
- **Output:** The number -18 is Negative



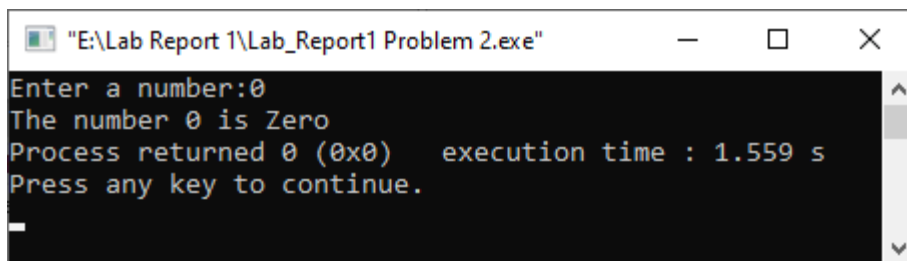
```
"E:\Lab Report 1\Lab_Report1 Problem 2.exe"
Enter a number:-18
The number -18 is Negative
Process returned 0 (0x0) execution time : 15.933 s
Press any key to continue.
```

- **Explanation:**

In this case, **num** is less than **0**. So, the program correctly displays the value of **num** as the negative number.

**Test Case 3:**

- **Input:** 0
- **Output:** The number 0 is Zero



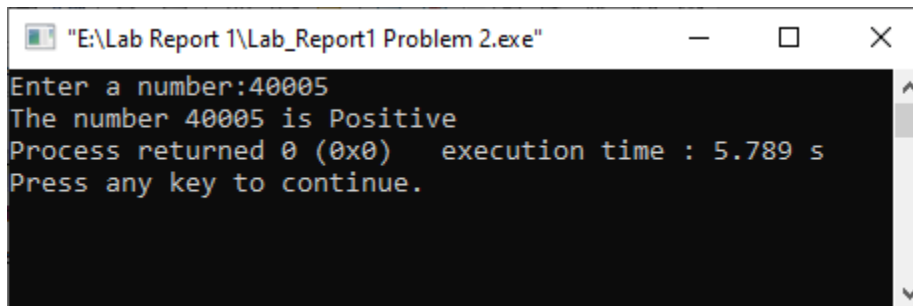
```
"E:\Lab Report 1\Lab_Report1 Problem 2.exe"
Enter a number:0
The number 0 is Zero
Process returned 0 (0x0) execution time : 1.559 s
Press any key to continue.
```

- **Explanation:**

In this case, **num** is equal to **0**. So, the program correctly shows the value of **num** as Zero.

**Test Case 4:**

- **Input:** 40005
- **Output:** The number 40005 is Positive



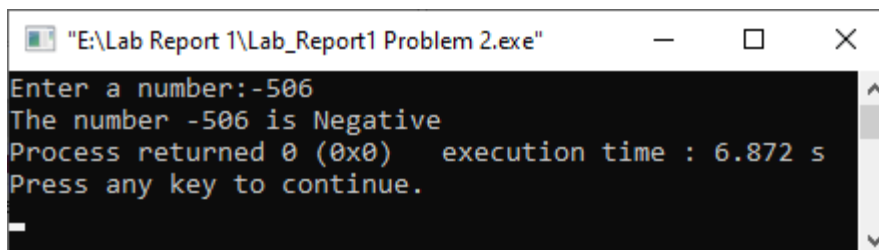
```
"E:\Lab Report 1\Lab_Report1 Problem 2.exe"
Enter a number:40005
The number 40005 is Positive
Process returned 0 (0x0) execution time : 5.789 s
Press any key to continue.
```

- **Explanation:**

In this case, **num** is greater than **0**. So, the program displays the value of **num** as the positive number.

**Test Case 5:**

- **Input:** -506
- **Output:** The number -506 is Negative



```
"E:\Lab Report 1\Lab_Report1 Problem 2.exe"
Enter a number:-506
The number -506 is Negative
Process returned 0 (0x0) execution time : 6.872 s
Press any key to continue.
```

- **Explanation:**

In this case, **num** is less than **0**. So, the program displays the value of **num** as the negative number.

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

## **7. Analysis and Discussions:**

Upon analyzing the program and its results, the following observations can be made:

- The program successfully determines whether a given number is negative, positive, or zero by implementing a C program.
- The implementation uses conditional statements to evaluate the input number and produce the appropriate output, which is an efficient approach for this task.

## **8. Summary:**

This lab helped strengthen the understanding of decision-making concepts of programming, input validation and displaying the result finally, while also providing hands-on experience in C programming.

## **Title: Lab Report 1.3**

### **Check whether a number is odd or even in C**

#### **1. Introduction:**

The objective of this lab is to develop a program in C programming language to determine whether a given number is odd or even. The program demonstrates the fundamental concept of decision-making and mathematical application in programming. Moreover, this lab involves implementing the necessary code, testing it with different inputs, analyzing the results, and discussing any observations or insights gained.

#### **2. Background:**

To determine the given number odd or even, it is needed to implement conditional statement. The algorithm compares the conditions and then displays the number whether to be odd or even as the output.

#### **3. Design:**

To determine the number odd or even in C, the following steps are used:

- i. Algorithm
- ii. Flowchart

The algorithm and flowchart created using the Lucid app are attached below:

#### Algorithm of a C program whether a number is odd or even

**Step 1:** Start

**Step 2:** Input num

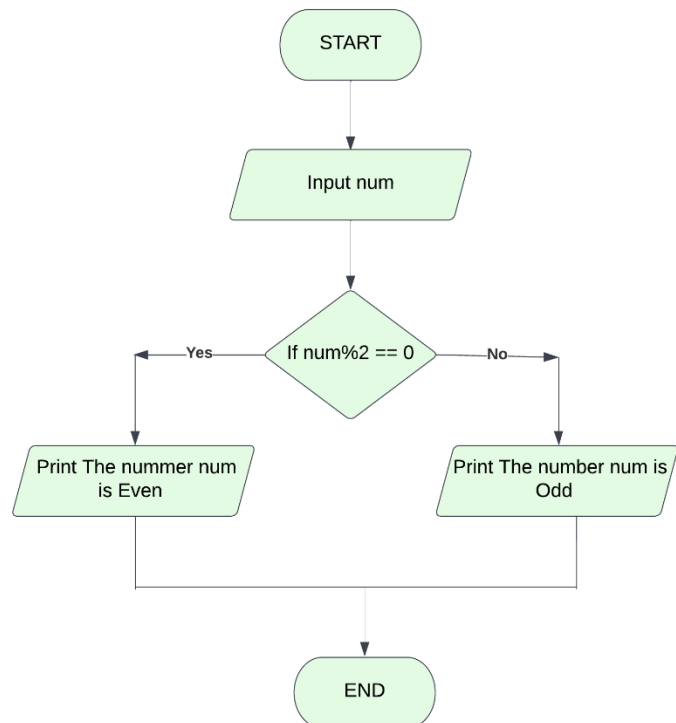
**Step 3:** If  $\text{num} \% 2 == 0$ , go to step 4, otherwise goto step 5

**Step 4:** Print The number num is Even, goto step 7

**Step 5:** Print The number num is Negative, go to step 8,

**Step 7:** End

#### Flowchart of a C program whether a number is is odd or even



## 4. Implementation:

```
1 // Write a C program to check whether a given number is odd or even.
2
3 #include<stdio.h>
4 int main ()
5 {
6     int num;
7
8     printf("Enter a number:");
9     scanf ("%d", &num);
10
11     if (num%2 == 0)
12     {
13         printf ("The number %d is Even", num);
14     }
15
16     else
17     {
18         printf ("The number %d is Odd", num);
19     }
20
21     return 0;
22 }
23
```

The program prompts the user to enter a number (num) using the scanf function. Then it checks the value of num using conditional statement. If the criteria of certain conditions of if statement or else statement meets, the program displays the number as odd or even on predefined condition using printf function.

## 5. Experimental Setup:

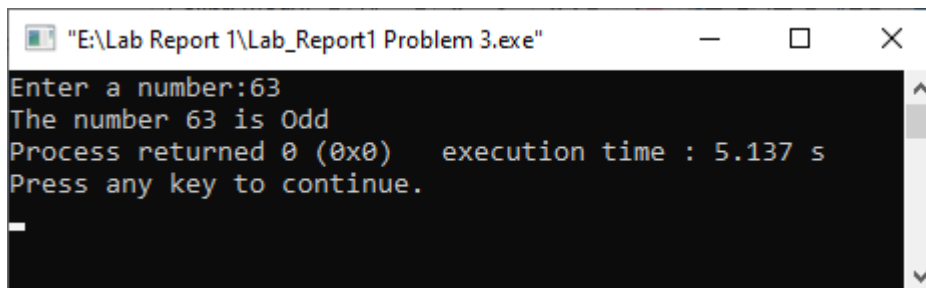
The program was implemented and tested using an Integrated Development Environment named "CodeBlocks" on a Microsoft operating system. The input for each test case was manually provided during the program execution.

## 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:** 63
- **Output:** The number 63 is Odd



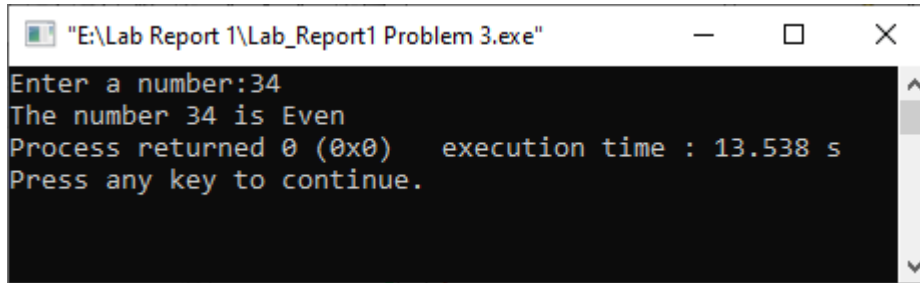
```
"E:\Lab Report 1\Lab_Report1 Problem 3.exe"
Enter a number:63
The number 63 is Odd
Process returned 0 (0x0) execution time : 5.137 s
Press any key to continue.
_
```

- **Explanation:**

In this case, **num** is divided by **2** having a remainder. So, the program correctly identifies 33 as the odd number.

### Test Case 2:

- **Input:** 34
- **Output:** The number 34 is Even



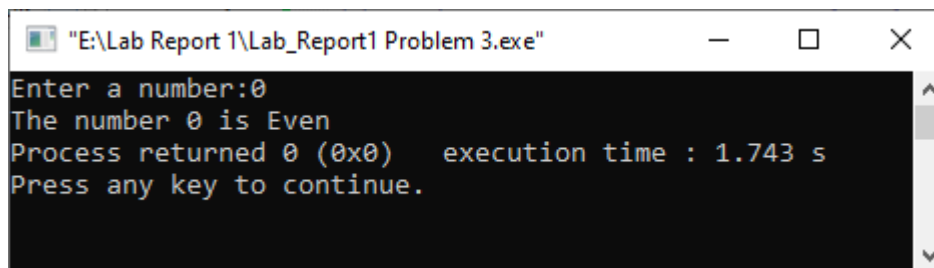
```
"E:\Lab Report 1\Lab_Report1 Problem 3.exe"
Enter a number:34
The number 34 is Even
Process returned 0 (0x0) execution time : 13.538 s
Press any key to continue.
```

- **Explanation:**

In this case, 34 (**num**) is divided by 2 without any remainder. So, the program correctly displays the value of **num** as the even number.

### Test Case 3:

- **Input:** 0
- **Output:** The number 0 is Even



```
"E:\Lab Report 1\Lab_Report1 Problem 3.exe"
Enter a number:0
The number 0 is Even
Process returned 0 (0x0) execution time : 1.743 s
Press any key to continue.
```

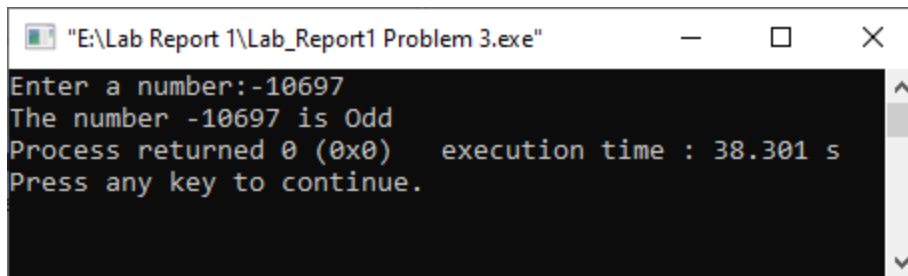
- **Explanation:**

In this case, **num** is also divided by 2 without any remainder. So, the program correctly shows the value of **num** as even.



#### Test Case 4:

- **Input:** -10697
- **Output:** The number -10697 is Odd



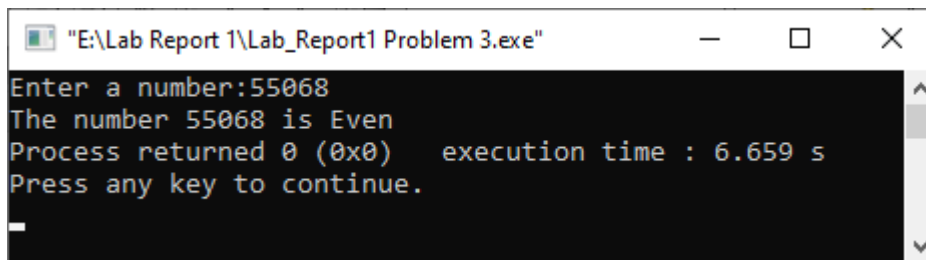
```
"E:\Lab Report 1\Lab_Report1 Problem 3.exe"
Enter a number:-10697
The number -10697 is Odd
Process returned 0 (0x0) execution time : 38.301 s
Press any key to continue.
```

- **Explanation:**

In this case, -10697 is divided by 2. But it provides a remainder value. So, the program displays the value of **num** as the odd number.

#### Test Case 5:

- **Input:** 55068
- **Output:** The number 55068 is Even



```
"E:\Lab Report 1\Lab_Report1 Problem 3.exe"
Enter a number:55068
The number 55068 is Even
Process returned 0 (0x0) execution time : 6.659 s
Press any key to continue.
```

- **Explanation:**

In this case, the result of **num** mod 2 is zero. So, the program displays the value of **num** as an even number.

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

## **7. Analysis and Discussions:**

Upon analyzing the program and its results, the following observations can be made:

- The program accurately determines whether a given number is odd or even by checking the remainder of the number when divided by 2.
- The implementation uses a conditional statement to evaluate the remainder and produce the appropriate output, which is an efficient approach for this task.
- The modulo operator (%) is a key component of this program, and its behavior may vary in different programming languages. It is important to understand the specific behavior of the modulo operator in the chosen programming language to ensure accurate results.

## **8. Summary:**

The lab was successfully completed by implementing a C program to determine whether a given number is odd or even. In summary, the importance of this program lies in its ability to classify numbers, its application of decision-making and conditional statements, its relevance in mathematics, its emphasis on input validation, its potential for code reusability, and its educational value for learning programming concepts.