Green University of Bangladesh
Department of Computer Science and Engineering

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

Lab Report 2
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: 07/06/23
Course Instructor: Saurav Chandra Das

Lab Report Evaluation	
Marks:	Signature:
Comment:	Date:

Lab Report 2.1

Title: Find the Maximum between three numbers

1. **Introduction:**

   The objective of this lab is to develop a program in C program that finds the maximum between two numbers which is a common problem in programming. The program will accept three integers as input from the user and display the largest number as the output.

2. **Background:**

   Finding the maximum between three numbers involves comparing the values of the numbers and determining which one is larger. This can be achieved using conditional statements and logical operator in C.

3. **Design:**

   To find the maximum between three 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 three numbers

Step 1: Start

Step 2: Input num1, num2, num3

Step 3: If num1>num2 and num1>num2 , max = num1 go to step 6, otherwise step 4

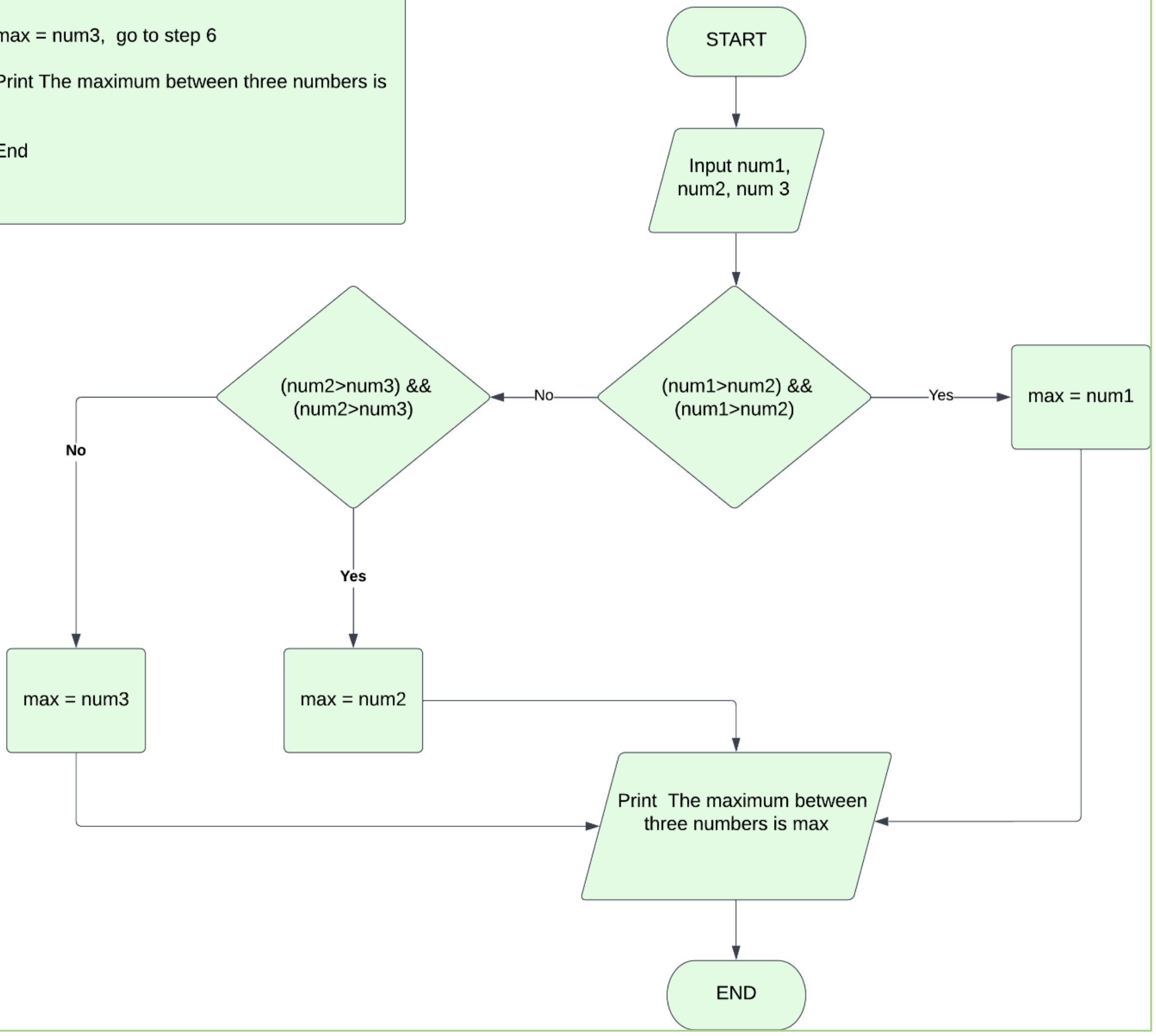
Step 4: If num2>num3 and num2>num3, max = num2, go to step 6; otherwise step 5

Step 5: max = num3, go to step 6

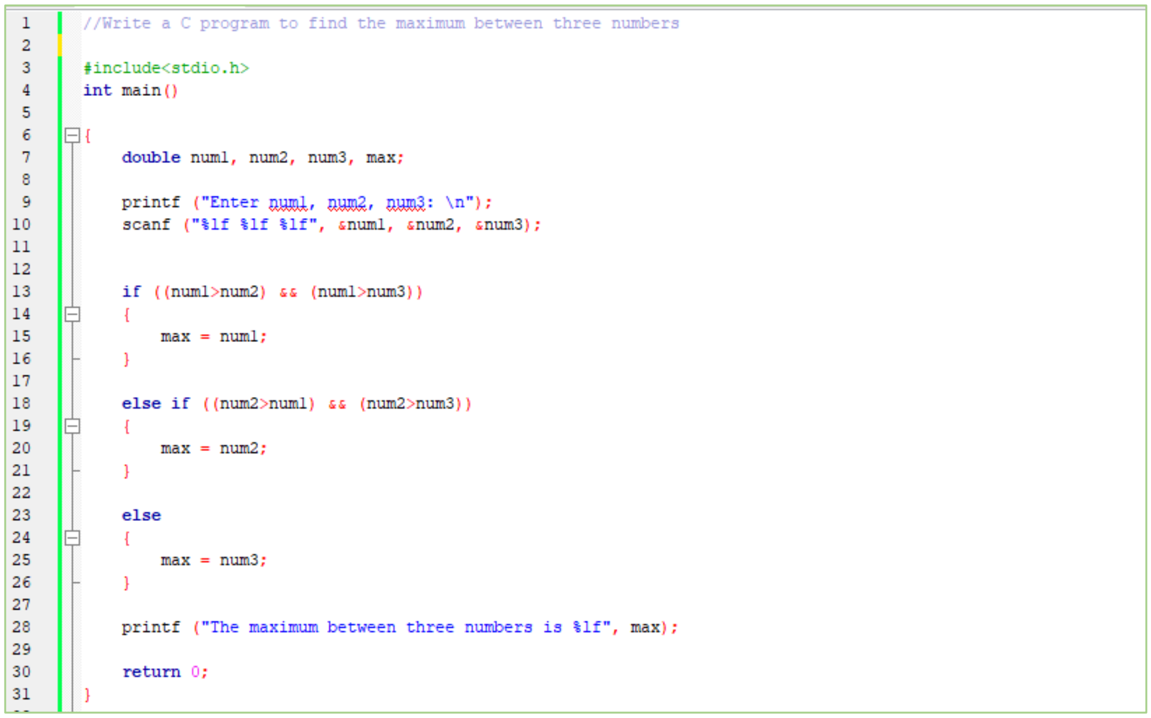
Step 6: Print The maximum between three numbers is max

Step 7: End

Flowchart of a C program whether a number is is odd or evenof a C program to find the maximum between three numbers


START
Input num1, num2, num 3
(num1>num2) && (num1>num2)
Yes
max = num1
No
(num2>num3) && (num2>num3)
No
Yes
max = num3
max = num2
Print The maximum between three numbers is max
END

4. Implementation:



```c
//Write a C program to find the maximum between three numbers

#include<stdio.h>
int main()

{
    double num1, num2, num3, max;

    printf ("Enter num1, num2, num3: \n");
    scanf ("%lf %lf %lf", &num1, &num2, &num3);


    if ((num1>num2) && (num1>num3))
    {
        max = num1;
    }

    else if ((num2>num1) && (num2>num3))
    {
        max = num2;
    }

    else
    {
        max = num3;
    }

    printf ("The maximum between three numbers is %lf", max);

    return 0;
}
```

The program prompts the user to enter three numbers using the scanf function. Then it compares the values of num1, num2 and num3 using an if-else if - else statement and logical operators. If num1 is greater than num2 and num 3, the max value will be num1. If num2 is greater than num1 and num3, the max value will be num2 otherwise the value of max will be num3. At last, this program display the maximum 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:

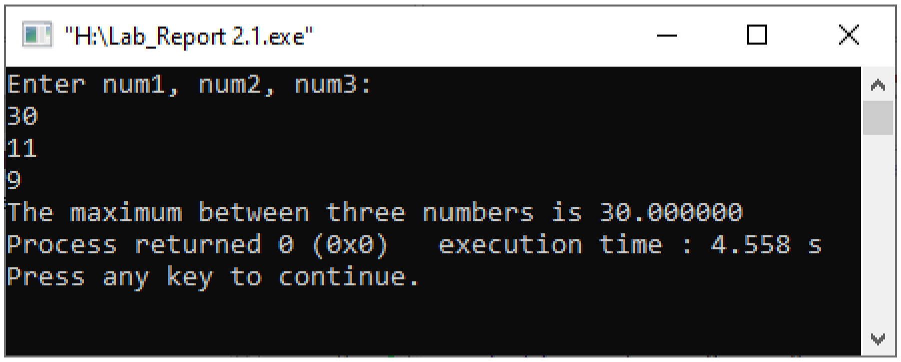
To ensure the correctness of the program, it has been conducted several test cases. Here are the results some:

Test Case 1:

- **Input:**

  The first number, **num1** is 30
  The second number, **num2** is 11
  The third number, **num3** is 9

- **Output:** The maximum number is 30



```
Enter num1, num2, num3:
30
11
9
The maximum between three numbers is 30.000000
Process returned 0 (0x0)   execution time : 4.558 s
Press any key to continue.
```

- **Explanation:**

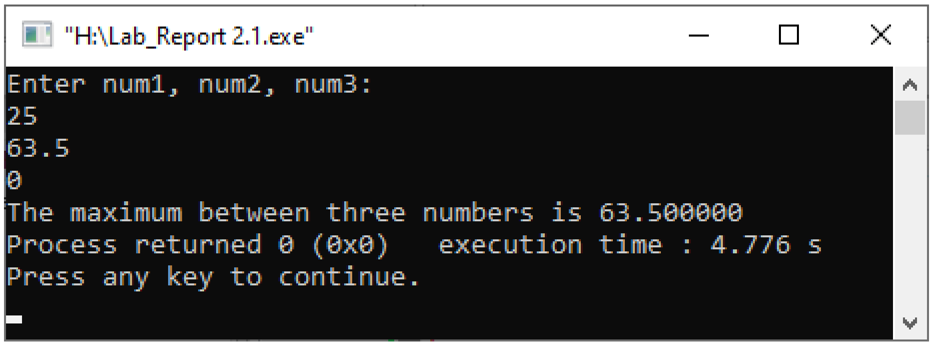
  In this case, **num1** is greater than both **num2** and **num3**. So, the program correctly identifies 23 as the maximum number.

Test Case 2:

- **Input:**

  num1 = 25
  num2 = 63.5
  num3 = 0

- **Output:** The maximum number is 63.5



"H:\Lab_Report 2.1.exe"

```
Enter num1, num2, num3:
25
63.5
0
The maximum between three numbers is 63.500000
Process returned 0 (0x0)   execution time : 4.776 s
Press any key to continue.
```

- **Explanation:**

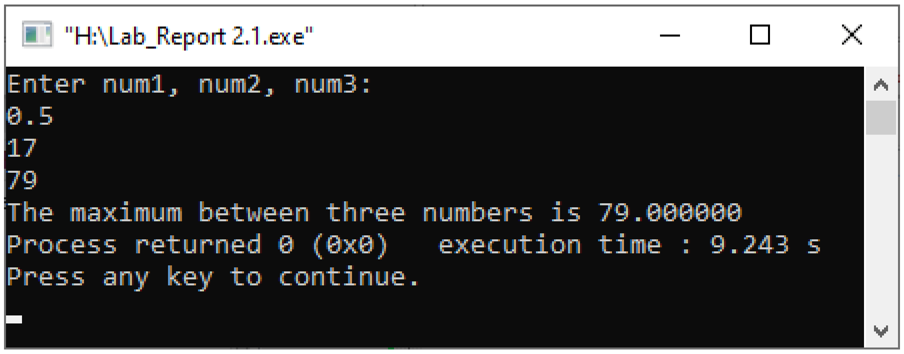
  In this case, **num2** is greater than **num1** and **num3**. So, the program correctly displays 63.5 as the maximum number.

Test Case 3:

- **Input:**

  num1 = 0.5
  num2 = 17
  num3 = 79

- **Output:** The maximum number is 79



"H:\Lab_Report 2.1.exe"

```
Enter num1, num2, num3:
0.5
17
79
The maximum between three numbers is 79.000000
Process returned 0 (0x0)   execution time : 9.243 s
Press any key to continue.
```

- **Explanation:**

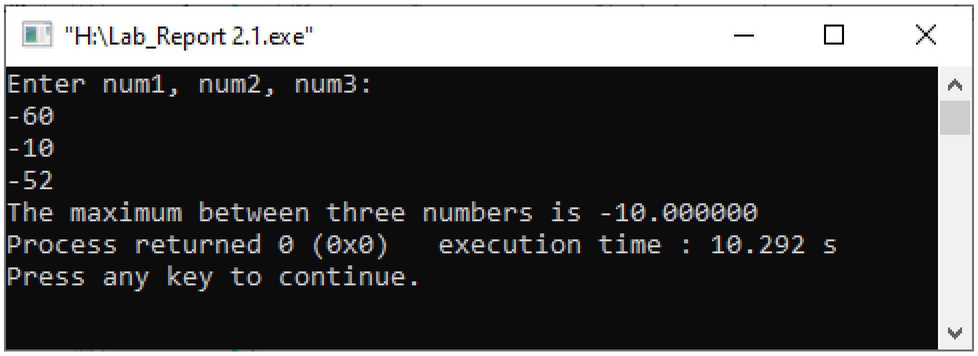
  In this case, 79 is greater than **num1** and **num2**. So, the program displays **num3** as the maximum number.

Test Case 4:

- **Input:**

  num1 = -60
  num2 = -10
  num3 = -52

- **Output:** The maximum number is -10


"H:\Lab_Report 2.1.exe"

```
Enter num1, num2, num3:
-60
-10
-52
The maximum between three numbers is -10.000000
Process returned 0 (0x0)   execution time : 10.292 s
Press any key to continue.
```

- **Explanation:**

  In this case, 79 is greater than **num1** and **num2**. So, the program displays **num3** as the maximum number.

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

7. Analysis and Discussions:

The program successfully achieves its objective of finding the maximum between three numbers. Overall, it is a simple yet crucial exercise that helps programmers develop fundamental skills and concepts that are applicable in more complex programming tasks. It serves as a building block for problem-solving and lays the groundwork for more advanced programming challenges. 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 important concepts of programming like problem-solving skills, algorithm thinking, understanding conditions, programming logic, validation and testing, while also providing hands-on experience in C programming.

Lab Report 2.2

Title: Take the value from the user as input for any alphabet and check whether it is vowel or consonant (using the switch statement)

1. Introduction:

The objective of this lab is to create a program in C programming language to determine whether a given alphabet is vowel or consonant using switch statement. The program will prompt the user to enter a single alphabet and display the corresponding output.

2. Background:

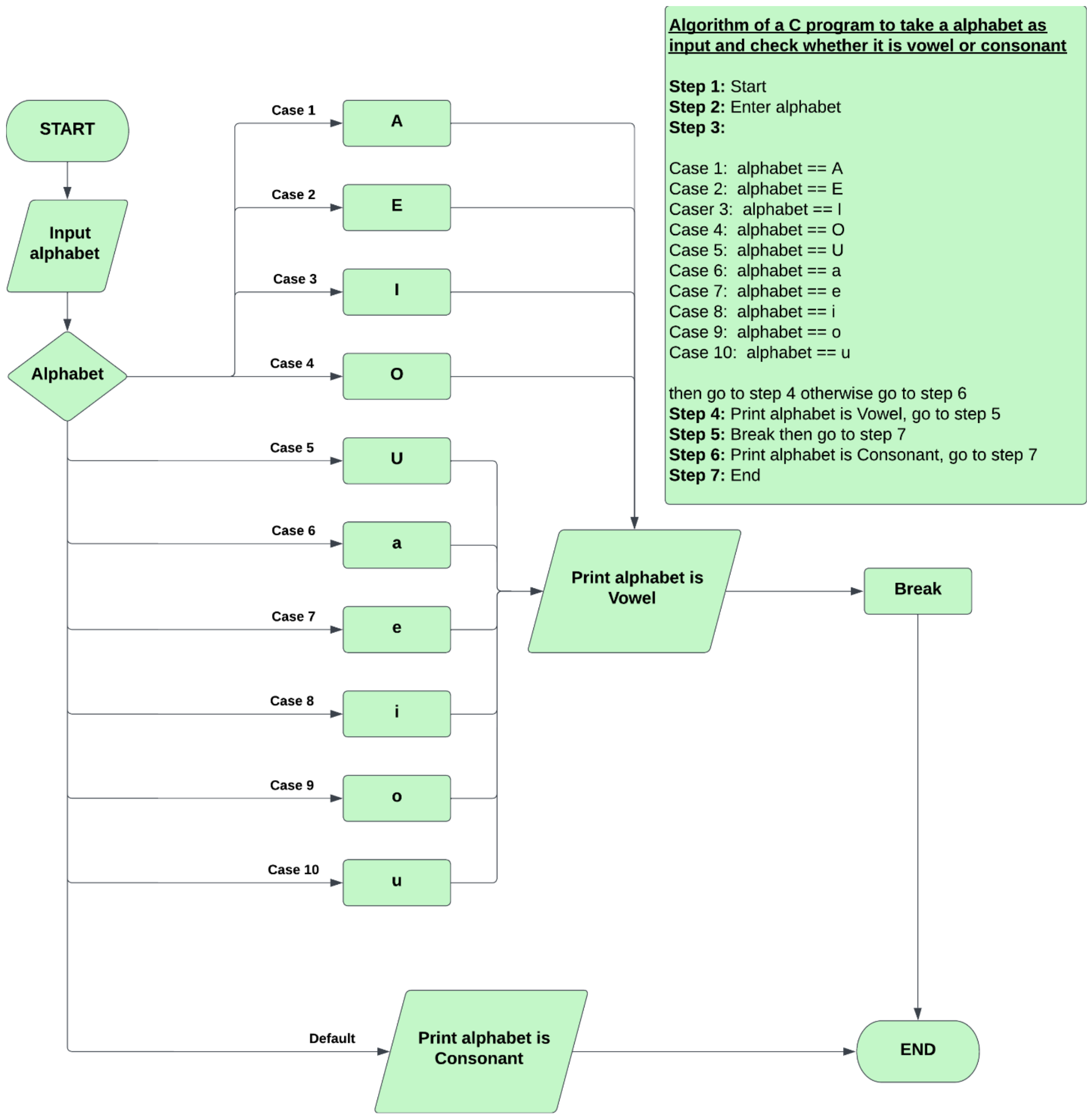
The task of determining whether an alphabet is a vowel or consonant is a common problem in programming. This can be achieved by using switch statement to check the input character against a set of predefined cases.

3. Design:

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

i. Algorithm
ii. Flowchart

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



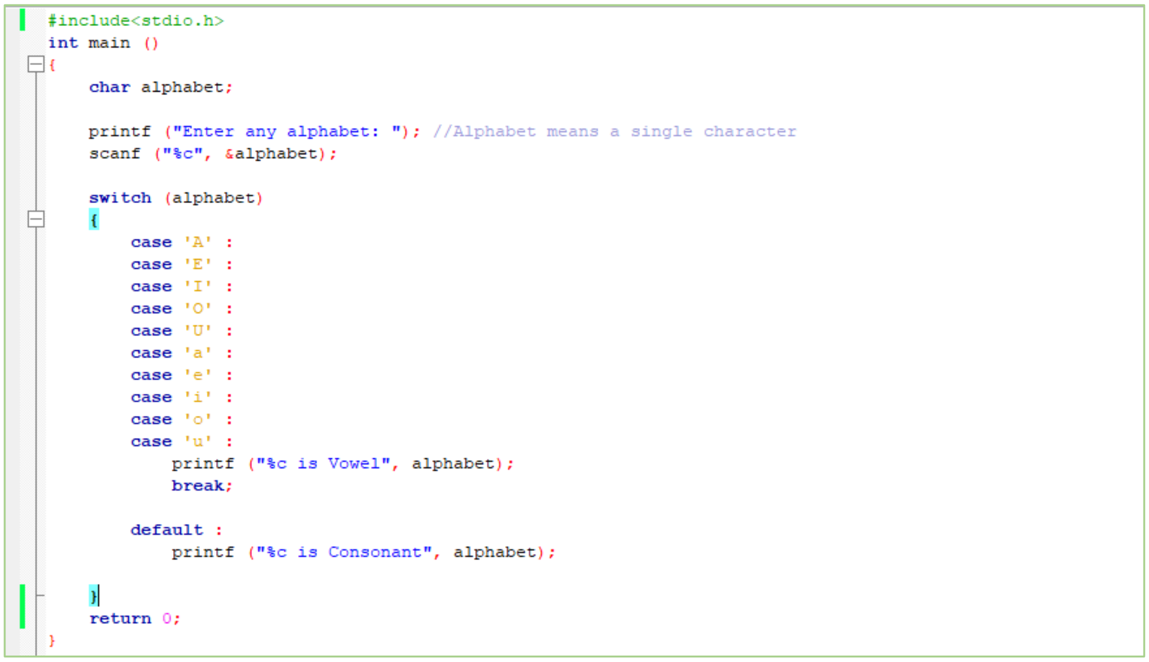
<u>Algorithm of a C program to take a alphabet as input and check whether it is vowel or consonant</u>

Step 1: Start
Step 2: Enter alphabet
Step 3:

Case 1: alphabet == A
Case 2: alphabet == E
Caser 3: alphabet == I
Case 4: alphabet == O
Case 5: alphabet == U
Case 6: alphabet == a
Case 7: alphabet == e
Case 8: alphabet == i
Case 9: alphabet == o
Case 10: alphabet == u

then go to step 4 otherwise go to step 6
Step 4: Print alphabet is Vowel, go to step 5
Step 5: Break then go to step 7
Step 6: Print alphabet is Consonant, go to step 7
Step 7: End

4. **Implementation:** The program prompts the user to enter an alphabet using the scanf function. Then it checks the value of the alphabet using swtich statement for equality against a list of values. The variable being switched on is checked for each switch case. If any switch case meets, the program displays its statement using printf function and then it encounters a break statement, otherwise it displays the statement of default.



```c
#include<stdio.h>
int main ()
{
    char alphabet;

    printf ("Enter any alphabet: "); //Alphabet means a single character
    scanf ("%c", &alphabet);

    switch (alphabet)
    {
        case 'A' :
        case 'E' :
        case 'I' :
        case 'O' :
        case 'U' :
        case 'a' :
        case 'e' :
        case 'i' :
        case 'o' :
        case 'u' :
            printf ("%c is Vowel", alphabet);
            break;

        default :
            printf ("%c is Consonant", alphabet);

    }
    return 0;
}
```

5. Experimental Setup:

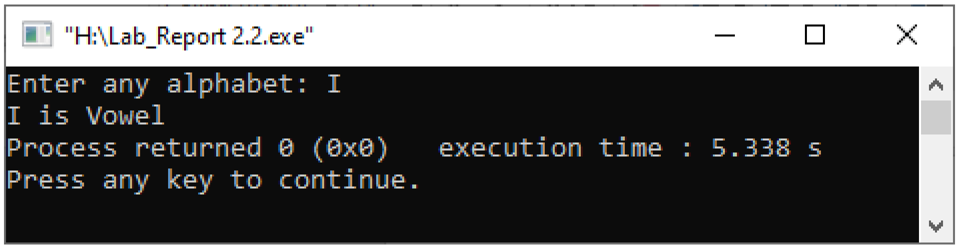
The program was implemented and tested using an Integrated Development Environment named “CodeBlocks” on a Microsoft operating system. The input for each test case was manually provided during the program execution.

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:** I
- **Output:** I is Vowel


"H:\Lab_Report 2.2.exe"

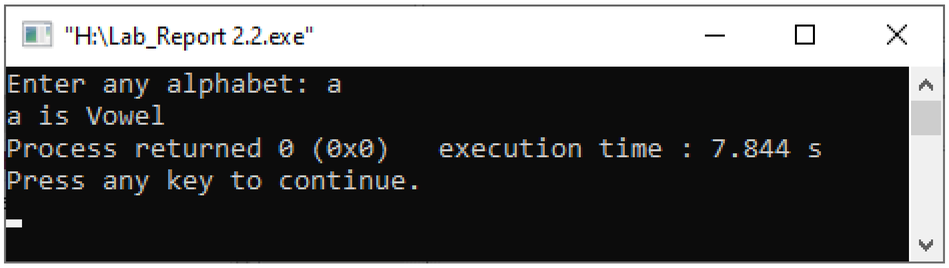
```
Enter any alphabet: I
I is Vowel
Process returned 0 (0x0)   execution time : 5.338 s
Press any key to continue.
```

- **Explanation:**

  In this case, **I** is matched with case 3. So, the program correctly identifies **I** as Vowel.

Test Case 2:

- **Input:** a
- **Output:** a is Vowel


"H:\Lab_Report 2.2.exe"

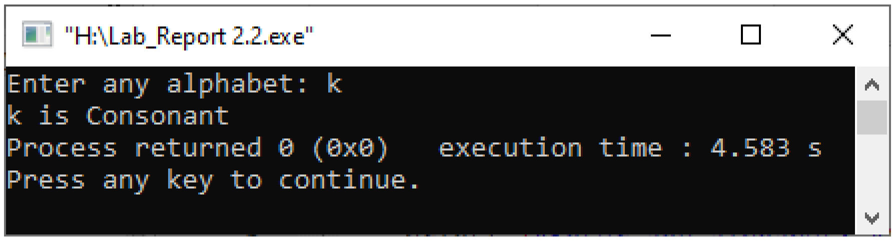
```
Enter any alphabet: a
a is Vowel
Process returned 0 (0x0)   execution time : 7.844 s
Press any key to continue.
```

- **Explanation:**

  Here, case 6 is matched with the value of the alphabet so the program correctly identifies it as Vowel.

Test Case 3:

- **Input:** k
- **Output:** k is Consonant



```
Enter any alphabet: k
k is Consonant
Process returned 0 (0x0)   execution time : 4.583 s
Press any key to continue.
```

- **Explanation:**

  In this case, no matching case clause is found, the program looks for the optional default clause, and displays its statement. So, the output is k is Consonant.

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 alphabet is vowel or consonant by implementing a C program.
- The implementation uses switch statements to evaluate the input alphabet and produce the appropriate output, which is an efficient approach for this task.

8. Summary:

This lab demonstrates the use of switch statements in C to solve a common problem and showcases the practical application of this programming construct.

Lab Report 2.3

Title: Check whether a year is a leap year or not in C

1. **Introduction:**

   The objective of this lab is to develop a program in C programming language to determine whether a given year is a leap year or not. The program will prompt the user to enter a year and display the corresponding output.

2. **Background:**

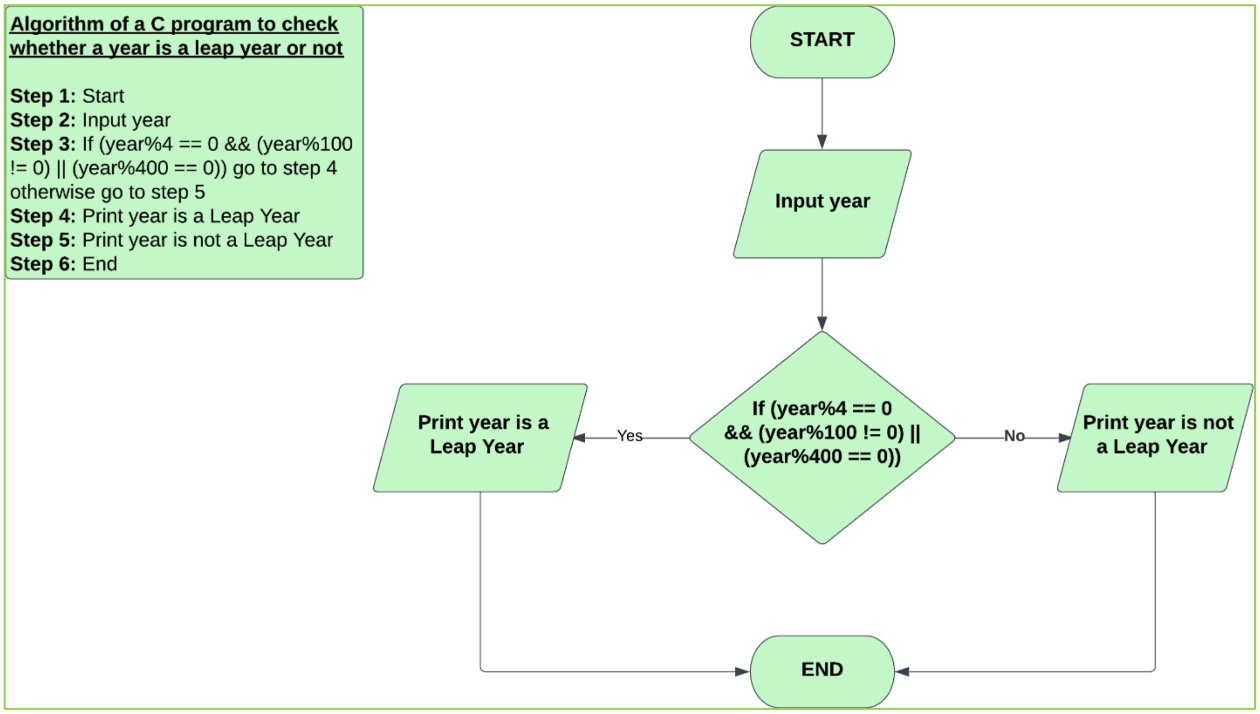
   To determine whether the given number leap year or not, it is needed to apply the necessary conditions and logical checks. The algorithm compares the conditions and then displays the output.

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:

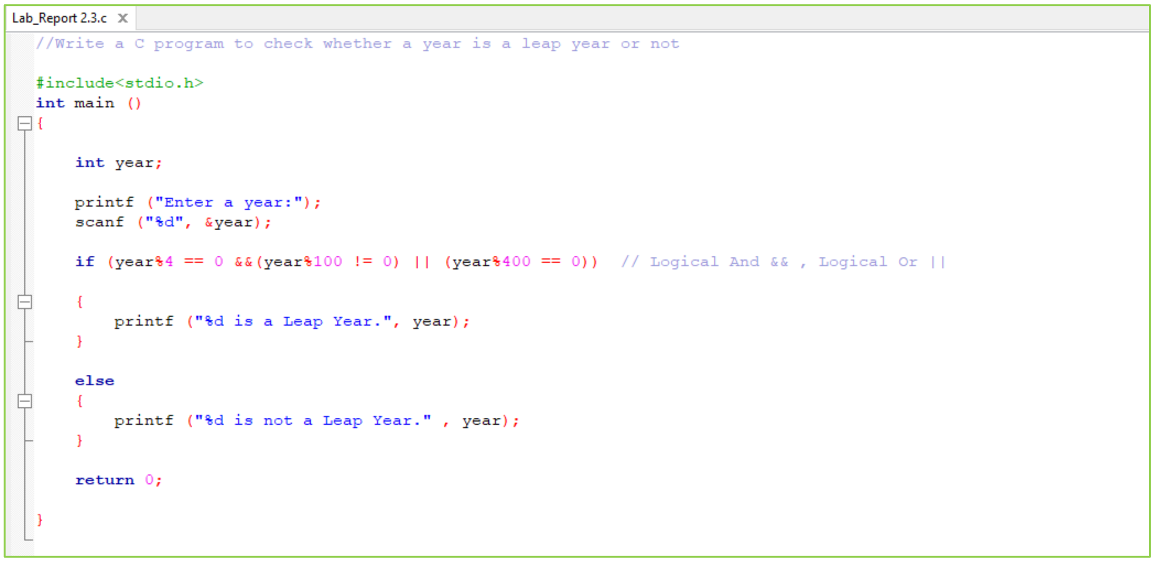


Algorithm of a C program to check whether a year is a leap year or not

```
Step 1: Start
Step 2: Input year
Step 3: If (year%4 == 0 && (year%100 != 0) || (year%400 == 0)) go to step 4 otherwise go to step 5
Step 4: Print year is a Leap Year
Step 5: Print year is not a Leap Year
Step 6: End
```

START
Input year
If (year%4 == 0 && (year%100 != 0) || (year%400 == 0))
Yes
No
Print year is a Leap Year
Print year is not a Leap Year
END

4. Implementation:



```c
//Write a C program to check whether a year is a leap year or not

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

    int year;

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

    if (year%4 == 0 &&(year%100 != 0) || (year%400 == 0))  // Logical And && , Logical Or ||

    {
        printf ("%d is a Leap Year.", year);
    }

    else
    {
        printf ("%d is not a Leap Year." , year);
    }

    return 0;

}
```

The program prompts the user to enter a year using the scanf function. Then it checks the value of year using conditional statement. If the criteria of certain conditions of if statement or else statement meets, the program displays the year as a leap year or not a leap year on predefined condition using printf function.

5. Experimental Setup:

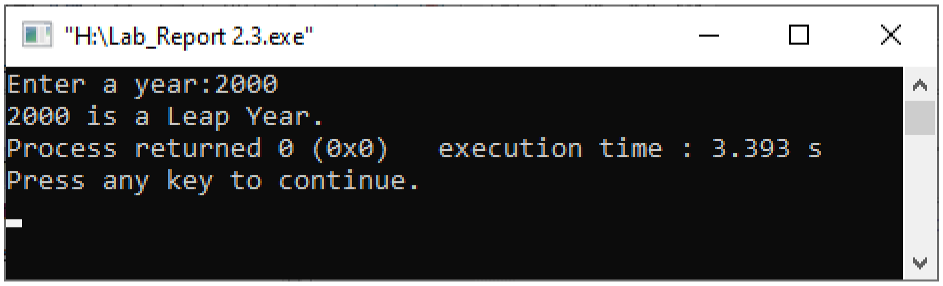
The program was implemented and tested using an Integrated Development Environment named “CodeBlocks” on a Microsoft operating system. The input for each test case was manually provided during the program execution.

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:** 2000
- **Output:** 2000 is a Leap Year.



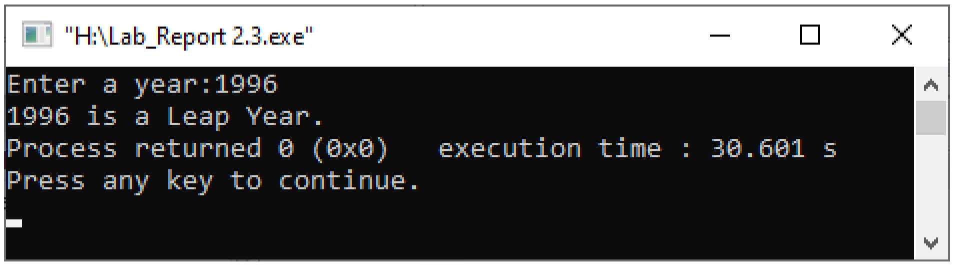
```
Enter a year:2000
2000 is a Leap Year.
Process returned 0 (0x0)   execution time : 3.393 s
Press any key to continue.
```

- **Explanation:**

  In this case, **year** is divided by **4** and **400** without any remainder. So, the program correctly identifies **2000** as leap year.

Test Case 2:

- **Input:** 1996
- **Output:** 1996 is a Leap Year.



"H:\Lab_Report 2.3.exe"

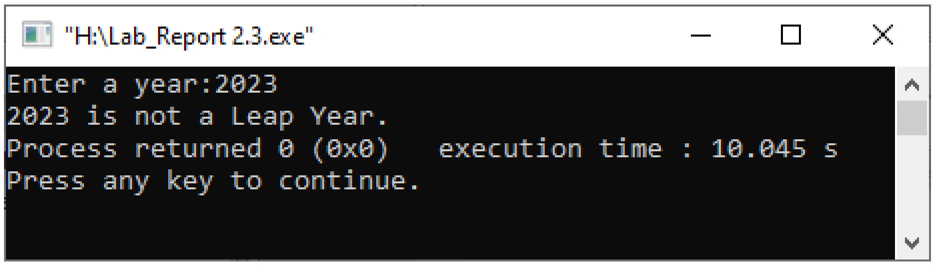
```
Enter a year:1996
1996 is a Leap Year.
Process returned 0 (0x0)   execution time : 30.601 s
Press any key to continue.
```

- **Explanation:**

  In this case, **year** is divided by **4** and **400** without any remainder. So, the program correctly identifies **1996** as leap year.

Test Case 3:

- **Input:** 2023
- **Output:** 2023 is not a Leap Year.



"H:\Lab_Report 2.3.exe"

```
Enter a year:2023
2023 is not a Leap Year.
Process returned 0 (0x0)   execution time : 10.045 s
Press any key to continue.
```

- **Explanation:**

  In this case, **year** is divided with having remainder. So, the program correctly identifies **2023** is not a leap year.

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 year is a leap year or not by checking the remainder of the year.
- The implementation uses a conditional statement to evaluate the divisibility of the year by 400, 100, and 4 and their remainder, 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.
- The implementation of the program is concise and efficient, requiring minimal lines of code.

8. Summary:

The lab was successfully completed by implementing a C program to determine whether a given year is a leap year or not. In summary, the importance of this program lies in its ability to classify years, 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.