DATE: LAB REPORT NO.: **1** SET: **A**
TITLE OF THE PROGRAM: **USER-DEFINED FUNCTIONS IN C**

OBJECTIVES

I. To understand the concept of "**No Return and No Argument**" user-defined functions in C programming.
II. To complete the following task using "**No Return and No Argument**" UDF.
    1. WAP to display the **Sum of two numbers**.
    2. WAP to display the **Area of the Rectangle**.
    3. WAP to check whether the number given by the user is **Positive, negative, or zero**.
    4. WAP to check whether the number given by the user is **Odd or even**.
    5. WAP to find the **greatest number** among the three numbers given by the user.
    6. WAP to find the **smallest number** among the three numbers given by the user.
    7. WAP to make **Calculator Program** using switch case.
    8. WAP to display the **multiplication table** for a number given by the user.
    9. WAP to check whether the number given by the user is **palindrome** or not.
    10. WAP to check whether the number given by the user is **Armstrong** or not.
    11. WAP to display the **factorial** of the number given by the user.
    12. WAP to display the **factors** for the number given by the user.
    13. WAP to display the **Fibonacci Series** for the nth term given by the user.
    14. WAP to check whether the number given by the user is **prime, or composite**.
III. To gain hands-on experience in writing modular and reusable code.
IV. To verify the correctness of the programs and analyze their Outputs.

REQUIREMENTS

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

User-defined functions in C allow programmers to create their own functions to perform specific tasks. These functions are separate from the main() function and can be called whenever required. They enhance code reusability, readability, and modularity. User-defined functions can be defined with or without arguments and can return values or perform actions without returning any value.

PROCEDURE (Program Code, Comment, and Output)

1. **Sum of Two Numbers:**

Program Code:

```c
#include <stdio.h>

void sumOfTwoNumbers(void); // Function prototype

int main()
{
    sumOfTwoNumbers(); // Function call

    return 0;
}
// Function Definition
void sumOfTwoNumbers(void)
{
    int num1 = 10, num2 = 20; // Initialize two numbers

    int sum = num1 + num2; // Calculate the sum

    // Display the result
    printf("Sum of %d and %d is: %d\n", num1, num2, sum);
}
```

Output:

```
Sum of 10 and 20 is: 30
```

2. **Area of a Rectangle:**

Program Code:

```c
#include <stdio.h>

// Function to calculate the area of a rectangle
void aor(void);

int main()
{
    // Call the aor function
    aor();
    return 0;
}

void aor(void)
```

```
{
    int length = 5, width = 10;

    // Calculate the area of the rectangle
    int area = length * width;

    // Display the calculated area
    printf("Area of the rectangle with length %d and width %d is:
%d\n", length, width, area);
}
```

Output:

```
Area of rectangle with length 5 and width 10 is: 50
```

3. **Checking Positive, Negative, or Zero:**

Program Code:

```
#include <stdio.h>

// Function to check if the number is positive, negative, or zero
void checkNumber(void);

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

void checkNumber(void)
{
    int number;

    // Prompt the user to enter a number
    printf("Enter a number: ");
    scanf("%d", &number);

    // Check if the number is positive, negative, or zero
    if (number > 0)
    {
        printf("%d is a positive number.\n", number);
    }
    else if (number < 0)
    {
        printf("%d is a negative number.\n", number);
    }
```

```
    else
    {
        printf("The number is zero.\n");
    }
}
```

Output:

```
Enter a number: 10
10 is a positive number.
```

4. **Checking Odd or Even:**

Program Code:

```
#include <stdio.h>

// Function to check if a number is odd or even
void checkOddEven(void);

int main()
{
    checkOddEven(); // Call the checkOddEven() function
    return 0;
}

void checkOddEven(void)
{
    int number;

    printf("Enter a number: ");
    scanf("%d", &number); // Input a number from the user

    if (number % 2 == 0)
    {
        // If the number is divisible by 2 without a remainder, it is
even
        printf("%d is an even number.\n", number);
    }
    else
    {
        // If the number is not divisible by 2 without a remainder, it
is odd
        printf("%d is an odd number.\n", number);
    }
}
```

Output:

```
Enter a number: 7
7 is an odd number.
```

5. **Finding the Greatest Number:**

Program Code:

```c
#include <stdio.h>

// Function to find the greatest number among three numbers
void findGreatestNumber(void);

int main()
{
    // Call the findGreatestNumber function
    findGreatestNumber();
    return 0;
}

void findGreatestNumber(void)
{
    // Declare variables to store three numbers
    int num1, num2, num3;

    // Prompt the user to enter three numbers
    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    // Check which number is the greatest
    if (num1 >= num2 && num1 >= num3)
    {
        // If num1 is the greatest
        printf("%d is the greatest number.\n", num1);
    }
    else if (num2 >= num1 && num2 >= num3)
    {
        // If num2 is the greatest
        printf("%d is the greatest number.\n", num2);
    }
    else
    {
        // If num3 is the greatest
        printf("%d is the greatest number.\n", num3);
```

```
    }
}
```

Output:

```
Enter three numbers: 10 25 15
25 is the greatest number.
```

6. **Finding the Smallest Number:**

Program Code:

```
#include <stdio.h>

// Function to find the smallest number among three numbers
void findSmallestNumber(void);

int main()
{
    findSmallestNumber(); // Call the findSmallestNumber function
    return 0;
}

void findSmallestNumber(void)
{
    int num1, num2, num3;

    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 <= num2 && num1 <= num3) // Check if num1 is the smallest
    {
        printf("%d is the smallest number.\n", num1);
    }
    else if (num2 <= num1 && num2 <= num3) // Check if num2 is the
smallest
    {
        printf("%d is the smallest number.\n", num2);
    }
    else
    { // num3 is the smallest
        printf("%d is the smallest number.\n", num3);
    }
}
```

Output:

```
Enter three numbers: 10 5 8
5 is the smallest number.
```

7. **Calculator Program using Switch Case:**

Program Code:

```c
#include <stdio.h>

// Function prototype
void calculator(void);

int main()
{
    // Call the calculator function
    calculator();
    return 0;
}

// Function to perform basic arithmetic calculations
void calculator(void)
{
    float num1, num2, result;
    char operator;

    // Prompt the user to enter two numbers
    printf("Enter two numbers: ");
    scanf("%f %f", &num1, &num2);

    // Prompt the user to enter an operator
    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &operator);

    // Perform the desired operation based on the operator
    switch (operator)
    {
        case '+':
            result = num1 + num2;
            printf("Result: %.2f\n", result);
            break;
        case '-':
            result = num1 - num2;
            printf("Result: %.2f\n", result);
            break;
```

```
        case '*':
            result = num1 * num2;
            printf("Result: %.2f\n", result);
            break;
        case '/':
            // Check if the second number is zero to avoid division by
zero error
            if (num2 != 0)
            {
                result = num1 / num2;
                printf("Result: %.2f\n", result);
            }
            else
            {
                printf("Error: Division by zero is not allowed.\n");
            }
            break;
        default:
            printf("Error: Invalid operator.\n");
    }
}
```

Output:

```
Enter two numbers: 10 5
Enter an operator (+, -, *, /): *
Result: 50.00
```

8. **Multiplication Table:**

Program Code:

```
#include <stdio.h>

// Function prototype
void multiplicationTable(void);

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

// Function to display the multiplication table of a number
void multiplicationTable(void)
{
```

```
    int number;

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

    printf("Multiplication Table for %d:\n", number);

    // Loop to calculate and display the multiplication table
    for (int i = 1; i <= 10; i++)
    {
        printf("%d x %d = %d\n", number, i, number * i);
    }
}
```

Output:

```
Enter a number: 5
Multiplication Table for 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

9. **Checking Palindrome:**

Program Code:

```
#include <stdio.h>

// Function to check if a number is palindrome
void checkPalindrome(void);

int main()
{
    checkPalindrome();
    return 0;
}
```

```c
void checkPalindrome(void)
{
    int number, reversedNumber = 0, remainder, originalNumber;

    // Prompting the user to enter a number
    printf("Enter a number: ");
    scanf("%d", &number);

    originalNumber = number;

    // Reversing the number
    while (number != 0)
    {
        remainder = number % 10;
        reversedNumber = reversedNumber * 10 + remainder;
        number /= 10;
    }

    // Checking if the original number is equal to the reversed number
    if (originalNumber == reversedNumber)
    {
        printf("%d is a palindrome number.\n", originalNumber);
    }
    else
    {
        printf("%d is not a palindrome number.\n", originalNumber);
    }
}
```

Output:

```
Enter a number: 12321
12321 is a palindrome number.
```

10. Checking Armstrong Number:

Program Code:

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

// Function to check if a number is an Armstrong number
void checkArmstrongNumber(void);

int main()
{
    checkArmstrongNumber();
```

```
    return 0;
}

void checkArmstrongNumber(void)
{
    int number, originalNumber, remainder, result = 0, digit_count=0;

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

    originalNumber = number;

    // Count digits in the number
    while(number!=0)
    {
        number=number/10;
        digit_count++;
    }

    number = originalNumber;

    // Calculating the sum of digits
    while (originalNumber != 0)
    {
        remainder = originalNumber % 10;
        result += pow(remainder,digit_count);
        originalNumber /= 10;
    }

    // Checking if the number is an Armstrong number
    if (result == number)
    {
        printf("%d is an Armstrong number.\n", number);
    }
    else
    {
        printf("%d is not an Armstrong number.\n", number);
    }
}
```

Output:

```
Enter a number: 153
153 is an Armstrong number.
```

11. Finding the Factorial:

Program Code:

```c
#include <stdio.h>

// Function to find factorial of a number
void findFactorial(void);

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

void findFactorial(void)
{
    int number;
    unsigned long long factorial = 1;

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

    // Check if the number is negative
    if (number < 0)
    {
        printf("Error: Factorial is not defined for negative
numbers.\n");
    }
    else
    {
        // Calculate the factorial of the number
        for (int i = 1; i <= number; i++)
        {
            factorial *= i;
        }

        // Print the factorial
        //the format specifier %llu for an unsigned long long
        printf("Factorial of %d is: %llu\n", number, factorial);
    }
}
```

Output:

```
Enter a number: 5
Factorial of 5 is: 120
```

12. Displaying Factors:

Program Code:

```c
#include <stdio.h>

// Function to display factors of a number
void displayFactors(void);

int main()
{
    // Call the displayFactors function
    displayFactors();
    return 0;
}

void displayFactors(void)
{
    int number;

    // Input the number from the user
    printf("Enter a number: ");
    scanf("%d", &number);

    // Print the heading for factors
    printf("Factors of %d: ", number);

    // Loop to find and display factors
    for (int i = 1; i <= number; i++)
    {
        // Check if 'i' is a factor of the number
        if (number % i == 0)
        {
            // Print the factor
            printf("%d ", i);
        }
    }

    printf("\n"); // Print a new line after displaying all factors
}
```

Output:

```
Enter a number: 12
Factors of 12: 1 2 3 4 6 12
```

13. Fibonacci Series:

Program Code:

```c
#include <stdio.h>

// Function to calculate and display Fibonacci series
void fibonacciSeries(void);

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

// Function definition
void fibonacciSeries(void)
{
    int num1 = 0, num2 = 1, nextTerm;
    int n;

    // Getting the number of terms from the user
    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");

    // Calculating and displaying the Fibonacci series
    for (int i = 1; i <= n; i++)
    {
        printf("%d, ", num1);
        nextTerm = num1 + num2;
        num1 = num2;
        num2 = nextTerm;
    }
}
```

Output:

```
Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

14. Checking Prime or Composite:

Program Code:

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

```c
// Function to check whether a number is prime or composite
void checkPrimeComposite(void);

int main()
{
    // Call the checkPrimeComposite function
    checkPrimeComposite();
    return 0;
}

void checkPrimeComposite(void)
{
    int number;
    int isPrime = 1;

    // Prompt the user to enter a number
    printf("Enter a number: ");
    scanf("%d", &number);

    // Check if the number is less than or equal to 1
    if (number <= 1)
    {
        printf("The number is neither prime nor composite.\n");
        return;
    }

    // Iterate from 2 to half of the number
    for (int i = 2; i <= number / 2; i++)
    {
        // Check if the number is divisible by i
        if (number % i == 0)
        {
            // If divisible, set isPrime to 0 and exit the loop
            isPrime = 0;
            break;
        }
    }

    // Check the value of isPrime
    if (isPrime)
    {
        printf("%d is a prime number.\n", number);
    }
    else
    {
```

```
        printf("%d is a composite number.\n", number);
    }
}
```

Output:

```
Enter a number: 7
7 is a prime number.
```

CONCLUSION

In this project, we successfully implemented user-defined functions in C programming to perform various mathematical operations. We achieved the objectives of understanding the concept of user-defined functions, developing modular code, and analyzing the outputs of the programs. Through this practical lab work, we gained hands-on experience in writing efficient and reusable code, enhancing our skills in C programming.