Experiment No. 4

CPL (C Programming Lab)

Aim : WAP to find all the prime numbers between two numbers using functions.

Software : Codeblocks & MingW

Theory : ***Functions***

A **function** in C is a block of code that performs a specific task. Functions help organize and reuse code, making programs more modular, easier to understand, and maintain.

Key Concepts of Functions:

1. **Function Declaration (Prototype)**: The function must be declared before it is used. A function declaration specifies the function's name, return type, and parameters.

return_type function_name(parameter_type1 param1, parameter_type2 param2, ...);

Example:

```
int add(int a, int b);
```

2. **Function Definition**: This is where the actual logic of the function is implemented. It includes the function header (same as the declaration) and the body, which contains the statements to be executed.

```
return_type function_name(parameter_type1 param1, parameter_type2 param2, ...) {
   // Function body (code to perform the task)
   return value;  // If the function has a return type other than `void`
}
```

Example:

```
int add(int a, int b) {
   return a + b;
}
```

3. **Function Call**: A function is called or invoked by writing its name followed by parentheses containing arguments, if any. The arguments passed to the function must match the parameters in the function definition in type and number.

```
function_name(argument1, argument2, ...);
```

Example:

```
int result = add(5, 3);
```

Types of Functions in C:

1. **Standard Library Functions**: Functions that are provided by C's standard library, such as printf(), scanf(), sqrt(), etc.
2. **User-Defined Functions**: Functions created by the programmer to perform specific tasks.

Key Components:

- **Return Type**: Specifies the type of value the function will return. It can be any valid C data type like int, float, char, or void (if no value is returned).
- **Function Name**: Identifier by which the function is called. It should follow C's naming rules.
- **Parameters (Arguments)**: Inputs to the function. The parameters are specified within the parentheses. If no parameters are needed, the parentheses are left empty.
- **Return Statement**: If the function has a return type other than void, it must include a return statement to send a value back to the calling function.

Example of a Function:

```
#include <stdio.h>

// Function declaration

int multiply(int x, int y);


int main() {

   int result = multiply(5, 3);  // Function call

   printf("Result: %d", result);

   return 0;

}


// Function definition
```

```
int multiply(int x, int y) {
    return x * y;
}
```

Advantages of Using Functions:

- **Modularity**: Functions break the program into smaller, manageable sections.
- **Reusability**: Once defined, a function can be used (called) multiple times in the program.
- **Maintainability**: Functions make code easier to maintain and debug.
- **Code Readability**: Functions enhance the structure and clarity of the code.

Post-Lab Questions:

1. How can you optimize the prime-checking function to make it faster?
2. How did you handle edge cases (e.g., when both numbers are the same or when the range includes negative numbers)?

Task 1: **WAP to find all the prime numbers between two numbers using functions. (Draw flowchart also)**

Program with Output:

Conclusion:

……

……

CO's Covered:

……

……