**LAB 11: Solving Real-World Problems Using C++ Programming Fundamentals**

**Objective**

The purpose of this assignment is to enhance students' ability to solve real-world problems using C++ programming. Each task focuses on **basic concepts**, including **switch statements**, **different types of loops**, **if-else statements**, **operators**, and **user-defined functions** (four types: **void functions, parameterized functions, value-returning functions, and functions with arrays**).

**Task Requirements**:

1. Each task must be implemented as a **C++ program** with 60-100 lines of code.
2. Modular programming using **user-defined functions** is mandatory.
3. No advanced function topics (e.g., recursion or templates) are allowed.
4. Ensure code readability and proper commenting.

**Tasks**

**Task 1: Traffic Light Simulation**

Design a program that simulates a **traffic light system**. The program will:

1. Ask the user to input the current light color (**Red**, **Yellow**, **Green**) using a **menu-driven system**.
2. Based on the input, display an action (e.g., Stop, Ready to Move, Go).
3. Allow the user to repeat the process for multiple inputs using a **do-while loop**.
4. Use a **switch statement** to handle light colors.

**Requirements**:

* Use a **void function** to display instructions for the traffic light system.
* Include a **switch statement** for determining actions based on light color.
* Use a **do-while loop** to repeat the process until the user chooses to exit.

**Task 2: Student Attendance Tracker**

A school requires a program to track student attendance and determine eligibility for exams. A student must attend at least **75% of classes** to be eligible.

**Requirements**:

1. Input the **total number of classes** and **number of classes attended** by the student.
2. Use a **value-returning function** to calculate the attendance percentage.
3. Use **if-else statements** to determine eligibility:
   * **Eligible**: Attendance ≥ 75%.
   * **Not Eligible**: Attendance < 75%.
4. Use a **loop** to allow multiple students’ data to be entered.
5. Display a summary at the end, showing the total number of students and how many were eligible.

**Task 3: Restaurant Menu System**

A restaurant wants a program to display a menu and take orders from customers. The program should:

1. Display a menu of items with their prices.
2. Use a **switch statement** to handle the selection of items.
3. Use a **parameterized function** to calculate the total price based on the selected item and quantity.
4. Use a **do-while loop** to allow customers to order multiple items.
5. Display the total bill at the end.

**Example Menu**:

* **1**: Pizza - $12
* **2**: Burger - $8
* **3**: Fries - $5
* **4**: Drinks - $3

**Task 4: Parking Fee Calculator**

Design a program to calculate parking fees for a parking lot. The fee structure is as follows:

* **Cars**: $10 per hour.
* **Motorcycles**: $5 per hour.
* **Bicycles**: $2 per hour.

**Requirements**:

1. Use a **menu-driven system** with a **switch statement** to select the type of vehicle.
2. Use a **parameterized function** to calculate the total parking fee based on the number of hours.
3. Use a **for loop** to calculate and display the parking fees for multiple vehicles.
4. Display the total earnings for the parking lot at the end of the program.

**Task 5: Grade Analysis System**

A school wants a program to analyze student grades and calculate the following:

1. The **highest grade** in the class.
2. The **average grade** of the class.
3. The **number of students who scored above 90%**.

**Requirements**:

1. Input grades for multiple students using an **array**.
2. Use a **function with arrays** to find the highest grade.
3. Use a **loop** to calculate the average and count students scoring above 90%.
4. Display all results in a well-formatted output.

**Submission Guidelines**

1. Submit your assignment as a **C++ source file** (.cpp).
2. Ensure that your code is well-commented for clarity.
3. Follow the modular programming approach with proper function definitions and calls.
4. Use meaningful variable names and proper indentation.

**Learning Outcomes**

1. Apply **basic programming concepts** to solve real-world problems.
2. Practice designing **menu-driven programs** and handling user input/output.
3. Develop logical thinking and problem-solving skills through **loops** and **decision-making constructs**.