Faculty of Computing

CS110: Fundamentals of Computer Programming

Class: BESE-16B

Lab 07: Function Arguments/Parameters, Variable Scope, Debugging Basics

CLO 1	**Understand** the syntax and semantics of different programming constructs
CLO 2	**Solve** given real-world problem by applying appropriate programming concepts and techniques

Date: 21st October 2025

Time: 2:00pm-5:00pm

Instructor: Dr. Momina Moetesum
Lab Engineer: Mr. Nadeem Nawaz

Introduction:
This lab introduces students with different concepts related to functions in C++. Students will learn how to pass parameters by value and by reference in functions and will also learn to create their own functions. By working through guided exercises, students will practice applying these constructs to solve practical problems.

Learning Objectives:
After completing this section, you will be able to:

- **Apply** modular constructs to solve real-world problems by implementing C++ programs.
- **Understand** debugging concepts by debugging an erraneous C++ program.

Tools/Software Requirement:

- Microsoft Visual Studio (any version)
- C++ Compiler (integrated within Visual Studio)
- Word Processor (MS Word or equivalent for compiling deliverables)

Task 1 [CLO 2]: Bank Loan Payment Calculator [Pass by Value + Default Parameters]

Apply the concepts of pass by value and default parameters to create a function. The function should take loan amount, annual interest rate, and number of payments (default 12 months). Use the formula:

$$P = [r*PV] / [1 - (1 + r)^(-n)]$$

where r is monthly rate, PV is present value, n is number of payments.
Write a C++ program that uses the above function to calculate the monthly payment for a loan.The program should handle edge cases like zero or negative inputs. The main function should test different loan scenarios (at least 2).

Task 2 [CLO 2]: Celsius-Fahrenheit Converter [Pass by Reference]

Use pass by reference function to write a C++ program that adjusts the temperature between Celsius and Fahrenheit scales. The function should take a temperature value and a boolean flag (Celsius to Fahrenheit or vice versa) and modify the temperature in-place.

Conversion formulas:
C to F: F = C * 9/5 + 32;
F to C: C = (F - 32) * 5/9.

Handle edge cases like extreme temperatures. The main function should demonstrate both conversions.

Task 3 [CLO 1]: Inline Functions

Understand how inline functions work by writing a C++ program that calculates the gravitational force between two objects using an inline function. The function should take masses of two objects and distance between them, returning the force.
Use the formula: F = G * (m1 * m2) / r^2,
where G is 6.67430e-11 N·m²/kg².
Handle edge cases like zero distance or negative masses. The main function should test different scenarios [at least 2].

Task 4 [CLO 1]: Understanding scope of variables and debugging

Debug a C++ program that calculates the total cost of a shopping cart with tax and discount. The program uses global, local, and block scope variables to compute the final price. The function calculateFinalPrice takes item price and quantity, applies a global tax rate, and a discount within a block scope. There are bugs related to variable scope and initialization that cause incorrect outputs. Debug the program to ensure it correctly calculates:
total = (price * quantity) * (1 + taxRate) * (1 - discount).
The main function tests different scenarios. Identify and fix the bugs to get correct outputs.

```
#include <iostream>

// Global variable for tax rate
double taxRate = 0.08; // 8% tax rate

double calculateFinalPrice(double price, int quantity) {
    double total; // Bug 1: Uninitialized variable
    {
        double discount = 0.1; // 10% discount in block scope
        total = price * quantity; // Bug 2: Missing tax and discount
application
        double taxRate = 0.05; // Bug 3: Local taxRate shadows global
taxRate
    } // Bug 4: Discount variable goes out of scope, not used correctly

    std::cout << "Debug: Tax rate used: " << taxRate << std::endl; //
For debugging
    return total;
}

int main() {
    double price = 50.0;
    int quantity = 2;
    double expectedTotal = (price * quantity) * (1 + 0.08) * (1 - 0.1);
// Expected: 50 * 2 * 1.08 * 0.9 = 97.2
```

```
    double result = calculateFinalPrice(price, quantity);
    std::cout << "Total cost for " << quantity << " items at $" << price
              << ": $" << result << std::endl;

    // Test with different values
    price = 100.0;
    quantity = 1;
    expectedTotal = (price * quantity) * (1 + 0.08) * (1 - 0.1); //
Expected: 100 * 1 * 1.08 * 0.9 = 97.2
    result = calculateFinalPrice(price, quantity);
    std::cout << "Total cost for " << quantity << " items at $" << price
              << ": $" << result << std::endl;

    return 0;
}
```

Expected output for first test: $97.2
Expected output for second test: $97.2
Current buggy output will be incorrect due to scope and initialization issues.

Debugging Hints for Students:

1. Check if variables are properly initialized before use.
2. Verify if the correct tax rate is applied (global vs. local scope).
3. Ensure the discount is applied correctly within the block scope.
4. Confirm the calculation includes both tax and discount in the correct order.

Deliverables:

Compile a single pdf document as displayed in solution/answer part and submit this pdf file on LMS.

Lab Rubrics:

Your Lab 7 will be graded out of 5 for each rubric according to the following rubrics.

Lab Rubrics for Lab 7					
Sr. No.	**Assessment**	**Unacceptable (0 Marks)**	**Does Not Meet Expectations (1/2 Marks)**	**Meets Expectations (3/4 Marks)**	**Exceeds Expectations (5 Marks)**
1	**Illustrating the basic understanding of semantics and syntax (CLO1, PLO1)**	The student did not submit any work. OR The student plagiarized the solution and/or used unfair means.	The student is unable to demonstrate the understanding of syntax of C++ language and is unable to write an executable code. The student is not able to understand the structure of a program at all.	The student demonstrates some understanding of syntax of C++ language and is able to write a code with few errors. The student is able to understand the structure but still learning the syntax.	The student demonstrates good understanding of syntax of C++ language and is able to write executable code without help The student is able to understand the structure and is able to identify problems in the code when introduced.
2	**Application of Programming Concepts (CLO2, PLO3)**		The student is unable to apply the appropriate programming concepts to solve the given problem thus resulting in an incomplete or ineffective solution. The program flow is messy and incomprehensible. Codes are non-modular and cannot be reused.	The student requires some guidance to apply the appropriate programming concepts to solve the given problem. The program flow requires minor improvements. Codes are semi-modular and semi-reusable.	The student demonstrates a clear ability to apply the appropriate programming concepts to solve the given problem. The program flow is adequate. Codes are modular, reusable, and easily readable.