National University of Computer and Emerging Sciences

Programming Fundamentals Lab (CL1002)

Date: 05/12/2025

Course Instructor(s)

Mr. Huzaifa Jawad

Lab Final Exam (B)

Total Time: 120 minutes
Total Marks: 50
Total Questions: **04**

Semester: FL-2025
Campus: Karachi
Dept: Computer Science

Name:________________________ Roll No:______________________

CLO # 1: Design and implement modular programs using functions, recursion and pointer manipulation.

Q1. [15 marks] Time: 35Min

A city library needs a program to manage book inventory, calculate late fees, and optimize shelf organization.

Requirements:

1. **Create a recursive function** calculateTotalFine that calculates the total late fee for n days. The function should:
   - Take base fine per day and number of days as parameters
   - Apply recursive calculation where each day's fine increases by 3 units from the previous day (starting from 5 units on day 1)
   - Return total fine amount
2. **Implement pointer-based functions** for managing book inventory:
   - swapPosition: Takes two integer pointers representing book IDs and swaps them to reorganize shelf position
   - findOldestBook: Takes an array of publication years and its size, returns a pointer to the oldest book's year
   - reorganizeShelf: Takes an array of book IDs and size, reverses the array in-place using two pointers to rearrange books from newest to oldest
3. **Create a function pointer system** for different membership fee calculations:
   - Define a function pointer type that can point to functions taking two integers (months, base_fee) and returning an integer
   - Implement three membership functions: studentMembership, regularMembership, and premiumMembership
   - Write a calculateMembershipFee function that accepts a function pointer to use the appropriate fee calculation

Main Function Requirements:

- Declare an array of 5 books with their publication years
- Call the recursive function to calculate total fine for 10 days
- Use pointer functions to find the oldest book and reorganize shelf
- Demonstrate function pointer usage by calculating fees with different membership types
- Display all results with appropriate messages

CLO # 2: Manipulate and process text data using multi-dimensional arrays and string handling functions.

Q2. [10 marks] Time: 25Min

Design a system to manage sports tournament registrations where player names and team codes are stored in 2D character arrays.

Requirements:

1. **2D Array Management:**
   - Create a 2D character array to store player names (maximum 25 players, 50 characters each)
   - Create another 2D array for team codes (format: "TM001", "FB202", etc.)
   - Implement inputPlayers: Safely input player names using fgets, preventing buffer overflow
2. **String Processing Functions:**
   - countPlayersInTeam: Takes the 2D team array, number of entries, and a specific team code. Uses strcmp to count how many players are in that team
   - validateTeamCode: Checks if a team code follows the pattern (2 letters + 3 digits) without using regex
   - isAnagramName: Checks if any two player names are anagrams of each other (ignore case and spaces) using strlen and character counting
3. **String Manipulation:**
   - generateJerseyName: Takes first name and last name as separate strings, concatenates them with a hyphen using strncat (ensure no overflow), converts to uppercase
   - sortPlayerNames: Sorts the 2D array of player names alphabetically using strcmp and any simple sorting algorithm

Main Function Requirements:

- Initialize arrays for 5 players and their team assignments
- Input player data and team codes
- Find how many players are in team "TM001"
- Generate jersey names for all players
- Sort and display the player list
- Check for anagram names and report findings

CLO # 3: Design complex data structures and manage memory dynamically for scalable applications.

Q3 [10 marks] Time: 20Min

Create an order tracking system for a restaurant that needs to manage customer orders and delivery scheduling.

Structure Definition: Define a structure named Order that contains the following members: an integer order ID for unique identification, a character array of size 50 to store the customer's name, an integer for table number, and a floating-point number for the total amount. Within this structure, create a nested structure for delivery details that includes integers for day, month, year, and hour (using 24-hour format). After the nested structure, include a character array of size 50 to store the assigned delivery person's name.

Required Functions:

1. **Input and Display:**
   - recordOrder: Takes a pointer to an Order structure and inputs all details. Validates that delivery hour is between 10-22 (restaurant hours)
   - displayOrderInfo: Takes an Order structure by value and displays all information in a formatted manner
2. **Processing Functions:**
   - calculateTip: Takes an Order structure and returns tip amount (15% for amount > 500, 10% for amount > 200, 5% otherwise)
   - findUrgentDelivery: Takes an array of Order structures and size, returns the Order with the earliest delivery time (compare year, month, day, then hour)
3. **Modification Functions:**
   - updateBilling: Takes a pointer to Order structure and applies the tip to the total amount
   - rescheduleDelivery: Takes a pointer to Order and new delivery details, updates the delivery time

Main Function Requirements:

- Create an array of at least 3 Order structures
- Record all orders with validation
- Find and display the order with most urgent delivery
- Calculate and apply tips to all eligible orders
- Display final billing information for all orders
- Demonstrate rescheduling for at least one order

Q4 [15 marks] Time: 35Min

Develop a comprehensive performance tracking system for a company that can handle varying team sizes throughout the year.

Requirements:

1. **Dynamic Array Creation:**
   - createPerformanceSheet: Dynamically allocates memory for n employee ratings (floats). Initializes all ratings to -1 (indicating not yet evaluated). Returns pointer to the array

  - createProjectMatrix: Dynamically allocates a 2D array for employee project assignments (rows = employees, columns = projects). Returns int** pointer

2. **Memory Management:**
  - expandTeam: Takes the current rating array pointer, old size, and new size. Reallocates memory to accommodate new employees while preserving existing ratings. Use either realloc or manual reallocation
  - free2DProjects: Properly frees the 2D project matrix in the correct order (first free each row, then the row pointers)

3. **Statistical Analysis:**
  - calculateMetrics: Takes the ratings array, size, and three float pointers (for average, highest, and lowest). Calculates metrics excluding -1 values and stores results through pointers
  - findTopPerformers: Dynamically allocates and returns an array containing indices of employees with ratings >= 75

4. **Performance Processing:**
  - inputRatings: Takes the ratings array pointer and size, inputs ratings with validation (0-100)
  - adjustRatings: Takes ratings array and size, adds a bonus of 10 points to all ratings below 50 (maximum 100)

Main Function Requirements:

- Start by asking for initial team size (e.g., 15 employees)
- Dynamically allocate rating array and project matrix (15 employees × 20 projects)
- Input ratings for initial employees
- Simulate hiring 8 more employees mid-year (expand arrays)
- Input ratings for new employees
- Calculate and display team metrics
- Find and display top performer indices
- Apply rating adjustments where needed
- Properly free all dynamically allocated memory before program ends
- Use proper error checking for all malloc/realloc calls

Memory Management Notes:

- Check if malloc/realloc returns NULL
- Free memory in reverse order of allocation
- No memory leaks allowed - every malloc must have corresponding free