National University of Computer and Emerging Sciences

Programming Fundamentals Lab (CL1002)

Date: 05/12/2025

Course Instructor(s)

Mr. Huzaifa Jawad

Lab Final Exam (A)

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 smart home company needs a program to manage energy consumption across different appliances. The system must track power usage, calculate bills, and optimize energy distribution.

Requirements:

1. **Create a recursive function** calculateMonthlyBill that calculates the total electricity bill for n days. The function should:
   - Take base cost per unit and number of days as parameters
   - Apply recursive calculation where each day's consumption increases by 2 units from the previous day (starting from 10 units on day 1)
   - Return total cost
2. **Implement pointer-based functions** for managing appliance power ratings:
   - swapPriority: Takes two integer pointers representing appliance power ratings and swaps them to reorganize priority
   - findMaxConsumer: Takes an array of power consumptions and its size, returns a pointer to the highest consuming appliance
   - optimizePower: Takes an array of power values and size, reverses the array in-place using two pointers to redistribute power from high to low priority
3. **Create a function pointer system** for different billing calculations:
   - Define a function pointer type that can point to functions taking two integers (units, rate) and returning an integer
   - Implement three billing functions: residentialRate, commercialRate, and industrialRate
   - Write a calculateBill function that accepts a function pointer to use the appropriate rate calculation

Main Function Requirements:

- Declare an array of 5 appliances with their power consumptions
- Call the recursive function to calculate monthly bill for 7 days
- Use pointer functions to find the maximum consumer and optimize power distribution
- Demonstrate function pointer usage by calculating bills with different rate 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 course registrations where student names and course codes are stored in 2D character arrays.

Requirements:

1. **2D Array Management:**
   - Create a 2D character array to store student names (maximum 30 students, 50 characters each)
   - Create another 2D array for course codes (format: "CSE101", "MTH202", etc.)
   - Implement inputRegistrations: Safely input student names using fgets, preventing buffer overflow
2. **String Processing Functions:**
   - countStudentsInCourse: Takes the 2D course array, number of entries, and a specific course code. Uses strcmp to count how many students registered for that course
   - validateCourseCode: Checks if a course code follows the pattern (3 letters + 3 digits) without using regex
   - isPalindromeName: Checks if any student name is a palindrome (ignore case and spaces) using strlen and character comparison
3. **String Manipulation:**
   - generateUsername: Takes first name and last name as separate strings, concatenates them with an underscore using strncat (ensure no overflow), converts to lowercase
   - sortStudentNames: Sorts the 2D array of student names alphabetically using strcmp and any simple sorting algorithm

Main Function Requirements:

- Initialize arrays for 5 students and their course selections
- Input student data and course codes
- Find how many students registered for "CSE101"
- Generate usernames for all students
- Sort and display the student list
- Check for palindrome names and report findings

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

Q3 [10 marks] Time: 20Min

Create a patient record system for a hospital that needs to track patient information and appointment scheduling.

Structure Definition: Define a structure named Patient that contains the following members: an integer patient ID for unique identification, a character array of size 50 to store the patient's name, an integer for age, and a floating-point number for the bill amount. Within this structure, create a nested structure for appointment 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 doctor's name.

Required Functions:

1. **Input and Display:**
   - registerPatient: Takes a pointer to a Patient structure and inputs all details. Validates that appointment hour is between 8-17 (hospital hours)
   - displayPatientInfo: Takes a Patient structure by value and displays all information in a formatted manner
2. **Processing Functions:**
   - calculateDiscount: Takes a Patient structure and returns discount amount (20% for age > 60, 10% for age < 12, 0% otherwise)
   - findEarliestAppointment: Takes an array of Patient structures and size, returns the Patient with the earliest appointment (compare year, month, day, then hour)
3. **Modification Functions:**
   - updateBilling: Takes a pointer to Patient structure and applies the discount to the bill amount
   - rescheduleAppointment: Takes a pointer to Patient and new appointment details, updates the appointment

Main Function Requirements:

- Create an array of at least 3 Patient structures
- Register all patients with validation
- Find and display the patient with earliest appointment
- Calculate and apply discounts to all eligible patients
- Display final billing information for all patients
- Demonstrate rescheduling for at least one patient

Q4 [15 marks] Time: 35Min

Develop a comprehensive grade management system for a programming course that can handle varying class sizes throughout the semester.

Requirements:

1. **Dynamic Array Creation:**
   - createGradeSheet: Dynamically allocates memory for n student grades (floats). Initializes all grades to -1 (indicating not yet graded). Returns pointer to the array
   - createAttendanceMatrix: Dynamically allocates a 2D array for student attendance (rows = students, columns = days). Returns int** pointer

2. **Memory Management:**
   - expandClass: Takes the current grade array pointer, old size, and new size. Reallocates memory to accommodate new students while preserving existing grades. Use either realloc or manual reallocation
   - free2DAttendance: Properly frees the 2D attendance matrix in the correct order (first free each row, then the row pointers)
3. **Statistical Analysis:**
   - calculateStatistics: Takes the grades array, size, and three float pointers (for average, maximum, and minimum). Calculates statistics excluding -1 values and stores results through pointers
   - findPassingStudents: Dynamically allocates and returns an array containing indices of students with grades >= 50
4. **Grade Processing:**
   - inputGrades: Takes the grades array pointer and size, inputs grades with validation (0-100)
   - adjustGrades: Takes grades array and size, adds a curve of 5 points to all grades below 40 (maximum 100)

Main Function Requirements:

- Start by asking for initial class size (e.g., 20 students)
- Dynamically allocate grade array and attendance matrix (20 students × 30 days)
- Input grades for initial students
- Simulate adding 10 more students mid-semester (expand arrays)
- Input grades for new students
- Calculate and display class statistics
- Find and display passing student indices
- Apply grade curve 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