FEU INSTITUTE OF TECHNOLOGY

Department of Information Technology

CCS0007 – COMPUTER PROGRAMMING 2

[SECTION]

# Midterm Machine Problem

Grade

Submitted by:

**<Last name, First name, Middle Initial>**

Submitted to

**<Professor>**

Submitted on:

**<Date>**

**Machine Problem 1: Student Database Management System**

**Topic:** User-defined Functions, Arrays, Structures, Character and String Manipulation

**Problem Statement:** You are tasked with creating a simple Student Database Management System in C++ that allows users to store and manipulate student information. The program should utilize user-defined functions, arrays, character and string manipulation, as well as structures.

**Requirements:**

1. **Student Structure:** Create a structure named Student with the following members:
    - int studentID
    - char firstName[50]
    - char lastName[50]
    - float GPA
2. **Function Definitions:**
    - void addStudent(Student students[], int &numStudents)
        - Prompt the user to input details for a new student and store them in the array.
    - void displayStudents(const Student students[], int numStudents)
        - Display the details of all students in a tabular format.
    - float calculateAverageGPA(const Student students[], int numStudents)
        - Calculate and return the average GPA of all students.
    - void findStudentsByLastName(const Student students[], int numStudents, const char lastName[])
        - Display the details of students whose last name matches the input.
3. **Main Program:** Implement a menu-driven program to:
    - Add a student
    - Display all students
    - Calculate the average GPA
    - Find students by last name
    - Exit

---

**Machine Problem 2: Inventory Management System**

**Topic:** User-defined Functions, Arrays, Structures

**Problem Statement:** Create a program to manage an inventory system for a small store. The program should use a structure to store item details and implement functions for different operations.

**Requirements:**

1. **Item Structure:** Create a structure named Item with the following members:
    - int itemID
    - char itemName[50]
    - int quantity
    - float price
2. **Function Definitions:**
    - void addItem(Item items[], int &numItems)
        - Prompt the user to input details for a new item and store it in the array.
    - void displayInventory(const Item items[], int numItems)
        - Display all items in a tabular format.
    - float calculateTotalValue(const Item items[], int numItems)
        - Calculate and return the total value of all inventory items (quantity * price).
    - void updateItemQuantity(Item items[], int numItems, int itemID, int newQuantity)
        - Update the quantity of a specific item based on the provided itemID.
3. **Main Program:** Create a menu-driven program to:
    - Add an item
    - Display inventory
    - Calculate total value
    - Update item quantity
    - Exit

---

**Machine Problem 3: Palindrome Checker**

**Topic:** Character and String Manipulation

**Problem Statement:** Write a program to check if a given string is a palindrome. The program should ignore case and whitespace while performing the check.

**Requirements:**

1. **Function Definitions:**
    - void toLowercase(char str[])
        - Convert all characters of a string to lowercase.
    - void removeWhitespace(char str[])
        - Remove all spaces from a string.
    - bool isPalindrome(const char str[])
        - Return true if the string is a palindrome, otherwise return false.
2. **Main Program:**
    - Prompt the user to enter a string.
    - Use the defined functions to preprocess the string and check if it is a palindrome.
    - Display the result (e.g., "The string is a palindrome" or "The string is not a palindrome").

**Machine Problem 4: Employee Record System**

**Topic:** User-defined Functions, Arrays, Structures

**Problem Statement:** Develop a program to manage employee records for a company. The program should allow adding, displaying, and searching for employees.

**Requirements:**

1. **Employee Structure:** Create a structure named Employee with the following members:
    - int employeeID
    - char name[100]
    - float salary
2. **Function Definitions:**
    - void addEmployee(Employee employees[], int &numEmployees)
        - Add a new employee record.
    - void displayEmployees(const Employee employees[], int numEmployees)
        - Display all employee records.
    - void findEmployeeByID(const Employee employees[], int numEmployees, int employeeID)
        - Search for an employee by their ID and display their details.
    - float calculateAverageSalary(const Employee employees[], int numEmployees)
        - Calculate and return the average salary of all employees.
3. **Main Program:** Create a menu-driven program to:
    - Add an employee
    - Display all employees
    - Search for an employee by ID
    - Calculate average salary
    - Exit

**II. Submission** Paste the source code and include 4 screenshots of the program output here

**Rubrics**

| TRAIT | Exceptional | Acceptable | Amateur | Unsatisfactory |
|---|---|---|---|---|
| Specifications (20) | The program works and meets all of the specifications. | The program works and produces the correct results and displays them correctly. It also meets most of the other specifications. | The program produces correct results but does not display them correctly. | The program is producing incorrect results. |
| Readability (20) | The code is exceptionally well organized and very easy to follow. | The code is fairly easy to read. | The code is readable only by someone who knows what it is supposed to be doing. | The code is poorly organized and very difficult to read. |
| Reusability (20) | The code could be reused as a whole or each routine could be reused. | Most of the code could be reused in other programs. | Some parts of the code could be reused in other programs. | The code is not organized for reusability. |
| Documentation (30) | The documentation is well written and clearly explains what the code is accomplishing and how. | The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code. | The documentation is simply comments embedded in the code with some simple header comments separating routines. | The documentation is simply comments embedded in the code and does not help the reader understand the code. |
| Efficiency (10) | The code is extremely efficient without sacrificing readability and understanding. | The code is fairly efficient without sacrificing readability and understanding. | The code is brute force and unnecessarily long. | The code is huge and appears to be patched together. |