SE200L: Data Structures and Algorithms Lab
Midterm Exam (Fall 2024)

Name: ***Roll#:***

Instructions:

1. Rename the folder of code to your roll no.
2. You have to attempt this question on computer, submit soft copy of code.
3. Time allowed is the maximum time allowed to solve the paper.
4. It is a closed book and closed notes exam.
5. Every question has the marks mentioned in the right most column, so manage the time and answers according to mentioned numbers.
6. You should not be taking assumption without mentioning those.
7. **Anybody found cheating or helping any fellow during exam will get his/her paper cancelled immediately.**
8. **You are not suppose to connect to internet or use browsers during this exam duration, anyone found using browsers will get his/her paper cancelled immediately.**
9. **Ensure you have working IDE, and compilation environment, use the same practice of organising files as already shared in lab and class.**
10. **Code with compilation errors will be marked zero.**

CLOs:

1. Should be able to develop code/programs to solve problems and to evaluate them to measure and contrast time and space requirements for various algorithms and data structures
2. Should be able to build solutions using modern tools (Clion, Cygwin & other IDE's) and utilize versioning capabilities of tools like GitHub
3. Should be able to value collaborative and individual work for solving problems
4. Should be able to complete solutions to programming problems independently, by following guidelines and/or specifications

Time allowed: 120 Minutes **Maximum Marks: 100**

	Marks Distribution	
Question	**1**	**Total**
CLO	1	
Total Marks	**100**	**100**
Marks Obtained		

Teaching Team: Nadir Abbas & Usama Riaz

Name .... **Roll#** ....

Q1

Student Aggregate Calculation

100

Exam Overview

The objective of this exam is to implement a linked list structure to manage student information and calculate their overall aggregate scores based on various assessments. Students will learn how to manipulate linked lists, apply sorting algorithms, and perform aggregate calculations.

Learning Outcomes

By completing this exam, you will:

Implement a linked list to store student records.
Manage data for multiple students effectively.
Sort student marks using selection and bubble sort algorithms.
Calculate aggregates based on weighted assessments.

Aggregate Calculation Overview

Each student's aggregate score is determined by their performance in several assessments, each contributing a specific weight to the final score. The key components include:

Assignments: The best 4 scores from 5 assignments (each worth 100 marks).
Quizzes: The best 4 scores from 5 quizzes (each worth 10 marks).
Project: A score from the project (out of 100).
Mid Exam: A score from the mid-exam (out of 100).
Final Exam: The final exam consists of **5 questions**, each worth **30 marks**.

The questions have varying weightages in the final aggregate:

Question 1: 15% weightage

Question 2: 10% weightage

Question 3: 25% weightage

Question 4: 30% weightage

Question 5: 20% weightage

Weightage Breakdown for Aggregate

Assignments: 10% of the total aggregate

Quizzes: 15% of the total aggregate

Project: 15% of the total aggregate

Mid Exam: 25% of the total aggregate

Final Exam: 45% of the total aggregate

Students will implement the logic for calculating the aggregate based on these criteria.

Student Class 50

Attributes 2

The Student class represents an individual student and includes the following attributes:

Name: (string) The full name of the student.
Roll Number: (string) A unique identifier for the student.
Assignment Marks: (float array of size 5) Marks for each of the 5 assignments.
Quiz Marks: (float array of size 5) Marks for each of the 5 quizzes.
Project Marks: (float) Marks obtained in the project (out of 100).
Mid Exam Marks: (float) Marks obtained in the mid-exam (out of 100).
Final Exam Marks: (float array of size 5) Marks for each of the 5 final exam questions (out of 30).
Aggregate: (float) The calculated aggregate percentage based on all assessments.
Student* next;

Functions

The Student class should include the following functions:

Constructor: Initializes the student's attributes. 2
Student(string name, string rollNumber);

Input Marks: Allows the user to input marks for assignments, quizzes, project, mid-exam, and final exam. 4
void inputMarks();

Set Assignment Marks: Takes an array as input and assign value to assignment marks array. 2
void setAssignmentMarks(const float marks[5])

Set Quiz Marks: Takes an array as input and assign value to quiz marks array. 2
void setQuizMarks(const float marks[5])

Set Project Marks: Setter function for Project marks. 2
void setProjectMarks(float marks)

Set Mid Exam Marks: Setter function for Mid exam marks. 2
void setMidExamMarks(float marks)

Set Final Exam Marks: Takes an array as input and assign value to final exam marks array. 2
void setFinalExamMarks(const float marks[5])

Sort Assignments: Sorts the assignment marks using bubble Sort (ascending or descending choice) 10
void sortAssignments();

Sort Quizzes: Sorts the quiz marks using bubble Sort 10
void sortQuizzes();

Calculate Aggregate: Computes the overall aggregate by calling the sorting functions and calculating contributions based on assessments. 10
void calculateAggregate();

Get Aggregate: Returns the overall aggregate. 2
float getAggregate();

StudentLinkedList Class 50

Attributes

The StudentLinkedList class manages a linked list of Student nodes and includes the following attributes:

Head: (Student*) A pointer to the first node (student) in the linked list.

Tail: (Student*) A pointer to the last node (student) in the linked list.

Count: (int) Maintains count of the student nodes.

Functions: The StudentLinkedList class should include the following functions:

Add Student:
Function: void addStudent(string name, string roll_number) 5
Description: Creates a new student node and adds it to the end of the linked list.

Input Student Marks:
Function: void inputMarks(string roll_number) 10
Description: Call the inputMarks method of student node having roll_number as parameter.
Function: void inputMarksforAll()
Description: Call the inputMarks method of each student in the linked list.

Get Student by Roll Number: 5
Function: Student* getStudent(string rollNumber)
Description: Takes roll_number as input and returns the student node address.

Calculate Aggregate for All Students:
Function: void calculateAggregates() 10
Description: Calls the calculateAggregate method for each student in the linked list.
This method will:
Call sortAssignments to sort the student's assignment marks.
Call sortQuizzes to sort the student's quiz marks.
Calculate the aggregate based on the sorted marks and other scores.

Sort Students by Aggregate: 10
Function: void sortStudentsByAggregate()
Description: Creates a temporary array of student node pointers, sorts the array by aggregate using selection sort, and reconstructs the linked list.

Display Students:
Function: void displayStudents() 5
Description: Prints the details of each student, including their name, roll number, and aggregate.

Get Aggregate of a Student: 5
Function: float getAggregate(string roll_number)
Description: It returns the aggregate of the required roll number.

Implementation Steps

Define the Student Node and Linked List:
Implement the Student and StudentLinkedList classes with their respective attributes and functions.
Input Student Data:
Use the addStudent and inputMarks functions to gather data for multiple students.
Calculate Aggregates:
Call calculateAggregates to compute the overall aggregate for each student, ensuring it utilizes the sorting functions from the Student class.
Sort Students by Aggregate:
Use the sortStudentsByAggregate function to sort the linked list based on aggregate scores.
Display Results:
Finally, utilize the displayStudents function to show each student's details and their aggregates.