Lab11

1. Lab Topics

This lab primarily covers multiple files, structs, and simple operator overload.

2. Updating the Teacher's Menu.

Remember the Teacher Menu in Lab11, in this lab, you are free to start working on a completely new files reusing codes from your previous work, but make sure that all the tasks assigned to this lab are fulfilled to obtain full grades. Follow the following steps to update the teacher's menu.

Step 1:

1. Create a struct named studentDetails to save the details of the student which will include studentID, firstName, lastName, course, year, major, score.
2. Create another struct named courseStatistics for saving the statistics for each course. It will include courseName, instructorName, higestScore, lowestScore, averageScore, and numStudents.

Use the appropriate data type to represent each of member variables. The type of studentID should follow prior work. numStudents represents the number of students taking that course.

Step 2:

1. Overload the extractor operator such that you can input the details for all the member variables of the struct studentDetails in a single call.
2. Overload the insertion operator such that you can output the details of a course statistics for all member variables in the struct courseStatistics in a single call. This is similar to the displayStatistics() function you have in your prior work.

Step 3:

1. In the main function, create ***n*** number of students for the studentDetails and call the extractor operator for all ***n*** students such that the member variables are populated. For Simplicity, you are free to create just 3 students.
2. Calling each of the member variables of the ***n*** students, find the highestScore, lowestScore, and averageScore.
3. Create a course with the courseStatistics struct and save the details in step 2 along with any courseName and instructorName of your choice to the courseStatistics.
4. Call the extractor operator to print out the courseStatistics in a good format.

BONUS (10):

1. For step 3.1, instead of creating 3 students with the studentDetails, ask the teacher for the number of students in the class. Use error exception to handle non integer data types

for the number of students in class. Then use a for loop to iterate through the number of students and call the overloaded extractor operator for each iteration. You can use an array for this purpose.

2. To obtain the class statistics, let your program only calculate the statistics for only students where the courseName is the same. i.e. check each of the student' courseName and select the students that have the exact match of courseName, and then use the score of the subset selected student to calculate the statistics for that course and return the courseName for name of the course in the struct course statistics.

Example output 1:

```
Enter student 0 details:
Enter studentID: ,./,,12
Enter firstName: Yu
Enter lastName: Mi
Enter courseName: CS1580
Enter graduation year: 2025
Enter Major: Comp_sci
Enter the score for the course: 89

Enter student 1 details:
Enter studentID: svds
Enter firstName: we
Enter lastName: wr
Enter courseName: CS1580
Enter graduation year: 2025
Enter Major: Comp_sci
Enter the score for the course: 56

Enter student 2 details:
Enter studentID: m,.wef
Enter firstName: Mi
Enter lastName: Yu
Enter courseName: Comp
Enter graduation year: 2025
Enter Major: Math
Enter the score for the course: 40

Course Statistics:
Course Name: Computer Science 101
Instructor Name: Dr. Smith
Highest Score: 89
Lowest Score: 40
Average Score: 61.6667
Number of Students: 3
```

3. How to get full marks

To get a 100% on this lab, follow this requirement:

1. Function Documentation and commenting of your program is extremely important (-20 marks).
2. Use multiple files in your program (-30 marks). Create a new folder to save the following file: make sure that only **THE REQUIRED** .cpp file is saved in this folder of yours.
    a. Use the header file (.h) to save your function prototypes and documentations,
    b. Use the header file (.hpp) to save your templated code (not covered in lab11),
    c. Use the main file (.cpp) to save your main function,
    d. Use the implementation file (.cpp) to save the implementation of your functions.
3. Follow the instructions as stated meeting all requirements for this lab (-50 marks).
4. Make sure your code compile and produce the necessary output before submitting.
5. Push your most recent code in git and submit through canvas as well. The canvas submission should include following files:
    a. The cpp file downloaded from git.
    b. **At least 1 Image files**
        i. **Screenshot of the right result similar to what I have in the example.**

Others

6. Use good variable names such that one can easily understand a variable's purpose just by looking at the name.
7. The program needs to be intuitive (e.g., display proper messages while you are taking user input or printing the result)
8. Follow all good coding conventions such as proper indentation.
9. Adhere to all coding standards outlined in lab2.
10. Follow the instructions of cloning, making dir, and submitting your code to git as previously discussed in lab01 and lab02.