CSCI 1110 – Fall 2024

Assignment 01

Please start this assignment early; programming and logic take time - if you leave it to the last minute, you might not have enough time to finish or might make silly mistakes that you otherwise could avoid. Note that TAs and Instructors will not be able to answer last-minute questions!

All work is to be handed in Codio, our online code-learning environment. However, you should write your code on an IDE such as IntelliJ.

To complete this assignment, you will need to know about:

- Basic Java
- Conditionals
- Boolean Variables
- Simple Input Validation
- Loops
- Arrays
- Methods

Your code **must compile**. If it does not compile, you will receive a 0 (zero) on that portion of the assignment, and no partial marks will be given.

Remember that students who ***hardcode*** their outputs to match the test cases in Codio will receive a **zero** on the entire assignment.

Grading Scheme: Please see the grading scheme at the end of this document.

Coding Style: You must use proper variable names and comments. Please follow the guidelines at https://web.cs.dal.ca/~franz/CodingStyle.html

You can create more methods than the ones described in the assignment.

Problem 1 – Calculating final marks (50 points)

In this assignment, we will write a program to calculate the final marks for a student in a course similar to CSCI 1110. In the first problem, we will calculate the final score for an individual student. In the next problems, we will compute class data.

For now, your program will read the students' banner number (String), test, assignments, practicums, and PoDs scores (int); they will always be in this order.

This class has a fixed number of grading assessments and weights:

1. Tests: 3 (25% of the final score)
2. Assignments: 5 (30% of the final score), drop lowest
3. Practicums: 4 (35% of the final score)
4. PoDs: 1 (10% of the final score)

You will calculate the final score as follows:

- Multiply the average of each grade item by its weight.
    - For the **assignment** grade item, you should **drop** the lowest score
- Sum each grade item together after multiplying them by their weights
- Convert the score into the final mark using the scoreToLetterGrade method

You must write the following static method:

- `string scoreToLetterGrade(double)`: This method will receive a grade as a numeric mark and convert to a letter grade. You must use the grade conversion found in https://www.dal.ca/campus_life/academic-support/grades-and-student-records/grade-scale-and-definitions.html

You can optionally implement the following static method:

- `double readAssessmentScores(int, boolean, Scanner)`: This method is optional (but highly recommended). It can be used to simplify reading input from the user. The method has three parameters: the number of grades it should read from the user (e.g.: 3 for tests), a boolean variable if the method should drop the lowest mark, a reference to the Scanner created in the main method.

After reading each assessment score and calculating the final mark, your program will print the student's banner number and the final mark.

Example:

Input	`B00123456 90 95 100 85 90 90 99 10 100 95 100 100 98`
Output	`B00123456 - A+`

Problem 2 – Multiple Students, Class Statistics (30 points)

Now, you will modify your program so it reads grades for multiple students in a class. You can copy your solution from P1 into P2 as a starter code. The program will calculate the final mark for each student in the same way as Problem 1.

Your program will first read the number of students in the class, then each student's information and print the student's banner and final mark. After that, your program will print the class averages of tests, assignments, practicums, and pods. You must limit the output to two decimal points.

For this problem, you **can**[1] use 1D or 2D Arrays; you **cannot use** ArrayLists or any other List collections. **If you solve this problem using lists or other collections, it will be graded as zero**

Example:

Input	3 B00123456 90 95 100 85 90 90 99 10 100 95 100 100 98 B00408996 30 12 45 40 90 30 45 60 55 65 32 80 50 B00989439 85 80 79 90 80 77 88 90 87 96 81 83 80
Output	B00123456 A+ B00408996 D B00989439 A- Class Averages: Tests: 68.44 Assignments: 78.92 Practicums: 81.17 PoDs: 76.00

[1] You can also solve this problem without arrays, lists, etc; it all depends on how you implement the solution

Problem 3 – Class' Median (20 Points)

In this last problem, you will calculate the median for each grade item. The median is defined as "a value in an ordered set of values below and above which there is an equal number of values or which is the arithmetic mean of the two middle values if there is no one middle number." The wiki has more info on it: https://en.wikipedia.org/wiki/Median

To calculate a median, you will need a sequence of numbers (in our case marks) ordered from the smallest to the largest (you can use `Arrays.sort`)

Your program will work the same way as problem 2. After it prints the average, it will then print the median for each grade item according to the example below.

For this problem, you must use Arrays. You **cannot use** ArrayLists or any other List collections. **If you solve this problem using lists or other collections, it will be graded as zero**

Example:

Input	3 B00123456 90 95 100 85 90 90 99 10 100 95 100 100 98 B00408996 30 12 45 40 90 30 45 60 55 65 32 80 50 B00989439 85 80 79 90 80 77 88 90 87 96 81 83 80
Output	B00123456 A+ B00408996 D B00989439 A- Class Averages: Tests: 68.44 Assignments: 78.92 Practicums: 81.17 PoDs: 76.00 Class Medians: Tests: 81.33 Assignments: 87.00 Practicums: 86.75 PoDs: 80.00

Grading Scheme

Each problem on the assignment will be graded based on three criteria:

Functionality

"Does it work according to specifications?" This is determined in an automated fashion by running your program on a number of inputs and ensuring that the outputs match the expected outputs. The score is determined based on the number of tests that your program passes.

Quality of Solution

"Is it a good solution?" This considers whether the solution is correct, efficient, covers boundary conditions, does not have any obvious bugs, etc. This is determined by visual inspection of the code. Initially full marks are given to each solution and marks are deducted based on faults found in the solution.

Code Clarity

"Is it well written?" This considers whether the solution is properly formatted, well-documented, and follows coding style guidelines (https://web.cs.dal.ca/~franz/CodingStyle.html). A single code clarity score is assigned for all solutions. If your program does not compile, it is considered non-functional and of extremely poor quality, meaning you will receive 0 for the solution.

	PROBLEM	PARTIAL POINTS	POINTS
PROBLEM 1			50
	Functionality	40	
	Quality of Solution & Code Clarity	10	
PROBLEM 2			30
	Functionality	20	
	Quality of Solution & Code Clarity	10	
PROBLEM 3			20
	Functionality	15	
	Quality of Solution & Code Clarity	5	
TOTAL		**100**	**100**