Exercise 1
Introduction to Classes and Objects

Overview

- This exercise is to be conducted by adopting a **Pair Programming.**
  What is pair programming? (https://youtu.be/oBraLLybGDA)
- You and your partner will be coding collaboratively using VS Code

Plagiarism Warning

You may discuss with others and refer to any resources. However, any kind of plagiarism will lead to your submission being dismissed. No appeal will be entertained at all.

Late Submission and Penalties

- The submission must be done via eLearning. Other than that (such as telegram, email, google drive, etc.), it will not be entertained at all.
- Programs that CANNOT COMPILE will get a 50% penalty.
- Programs that are submitted late will get a 10% penalty for every 10-minutes late.

Problem

In this exercise, you will be writing a C++ program that displays the grade earned for given score of a subject. You will be using an Object-Oriented Programming (OOP) approach to write the program.

Notes:

- Write the program in a single source code file and use the **inline style** to define the class. That means, the definition of each method is written directly in the class declaration.

- Follow proper naming convention:
  - Use camel Case to name functions, methods, and variables.
    - Example: `int calculateTotal(), int thisIsVariable`
  - Use Pascal Case to name class and data type.
    - Example: `class Student{ }`
  - Use CAPITAL case to all characters to name constants.
    - Example: `const int MAXIMUM_STUDENT = 10;`
  - Use small case to all characters to name a file
    - Example: `my_main_program.cpp`

Modify the starter program provided (main.cpp) to accomplish the following tasks:

1. Declare a class to model subject information which consists of three attributes, the subject's name, code, and the score earned for the subject. In your code, you should also implement the "data hiding" principle for the class.

2. Define two constructors for the class as follows:
   a. a **parameterized constructor** which accepts parameters to set the attributes.
   b. a **default constructor** method which sets the attributes to "empty" values

3. Define the **destructor** method for the class which does nothing.

4. Define an **accessor** method (also known as **getter**) and a **mutator** method (also known as **setter**) for **each attribute** of the class. For example, as for the attribute `code`, define two methods for it, i.e., **`getCode()`** and s**`etCode()`**. Keep in mind that the accessors should be declared as **constant methods**.

5. Define three more **accessor methods** below:
   a. a method that determines the grade earned for the subject based on the score. For example, if the score earned is `70`, this method should return `"B+"`.
   b. a method that determines the point value of the grade earned. For example, if the grade earned is `"B+"`, this method should return `3.33`.
   c. a method that determines the point earned for the subject. For example, if the subject is 3 credit hour and the grade point earned is `3.33` , this method should return `9.99`

   See the section below for the formulas.

6. In the main function, write the code to accomplish the following requirements:
   a. Instantiate or create an object from the class.
   b. Read inputs from the user and use them to set the attributes of the object.
   c. Print the subject information such as the code, name, score, grade, etc. See example runs in the following figures for the expected results.

   Example Run 1

```
Enter the following data:
  Subject name => Programming Technique II
  Subject code => SECJ1023
  Score earned => 95


THE RESULT

Subject Code : SECJ1023
Subject Name : Programming Technique II
Credit Hour  : 3
Score Earned : 95
Grade Earned : A+
Grade Point  : 4
Point Earned : 12
```

Example Run 2

```
Enter the following data:
  Subject name => Graduate Success Attributes
  Subject code => UHMT1012
  Score earned => 84


THE RESULT

Subject Code : UHMT1012
Subject Name : Graduate Success Attributes
Credit Hour  : 2
Score Earned : 84
Grade Earned : A
Grade Point  : 4
Point Earned : 8
```

Grade and Point Calculation and Formula

The grade and point value earned are determined based on the following table.

Score	Grade	Point Value
90 - 100	A+	4.00
80 - 89	A	4.00
75 - 79	A-	3.67
70 - 74	B+	3.33
65 - 69	B	3.00
60 - 64	B-	2.67
55 - 59	C+	2.33
50 - 54	C	2.00
45 - 49	C-	1.67
40 - 44	D+	1.33
35 - 39	D	1.00
30 - 34	D-	0.67
0 - 29	E	0.00

Credit hour is determined from the last digit of the course code. For example, the course with code SECJ1023 is 3 credit hours

Point Earned = Point value x credit

Assessment

This exercise carries **2.5%** weightage for the final grade of this course. The breakdown weightage is as follows (out of 100 points):

Criteria	Points
1. The code a. Task 1 – class declaration and class attributes b. Task 2 – constructors and destructor c. Task 3 – mutators and accessors (for the attributes) d. Task 4 – three more getters e. Task 5 – main function	 10 10 20 15 15
2. Pair Programming Session a. Overall b. Active collaboration c. Both members play both roles Driver and Navigator.	 10 10 10

Submission

- Deadline: at the end of the class
- Only one member from each pair needs to do the submission.
- Submission must be done on eLearning. Any other means such as email, telegram, google drive will not be accepted at all.
- Submit only the source code (i.e., main.cpp)

FAQs

1. **Who will be my partner?**

   You will choose your partner on your own.

2. **Can I do the exercise alone?**

   This is only allowed if the number of students in the class is not even. You also need to ask for permission from the lecturer.

3. **Do we need to switch roles between Driver and Navigator?**

   Yes. Your video should show that you and your partner keep switching between these two roles. No one should be dominant or play only one role.