CS381-L Software Engineering Lab 02

Type of Lab: Close Ended **Weightage: 5%**

CLO 2: Implement, in a programming language, an executable solution to a given problem using best practices.

Implement, in a programming language, an executable solution to a given problem using best practices.	**Cognitive/Apply**	CLO2	Rubric A

CLO 5: Comply with Plagiarism Policies.

Comply with Plagiarism Policies.	**Cognitive/Understand**	CLO2	Rubric B

Rubric A: Cognitive/Apply

CLO2	0	1	2	3	4
Implement, in a programming language, an executable solution to a given problem using best practices.	Student could not complete the task	Student understand the problem and solution but could not write the code	Student write code but with some minor errors	Student code is giving the desired results	Student code is correct with perfect indentation

Rubric B: Cognitive/Understand

CLO 5	**0**	**1**	**2**	**3**	**4**
Comply with plagiarism policies.	Students copied complete code from other fellows.	Student gives his code to other students.	Students partially copied or asked the code from other fellows.	Students copied the code but clearly mentioned the part of the code that has been taken.	Student did not copy the code and did not allow anyone to copy his code.

Lab 02

Assessment Criteria:

Implementation of Class Diagram: 4 points
GUI: 3 points
Validation: 1.5 points
Documentation Comments: 1.5 points

Assignment 2

Write a C# program in which you are required to implement the following design.
Constraints for each attribute are given below.

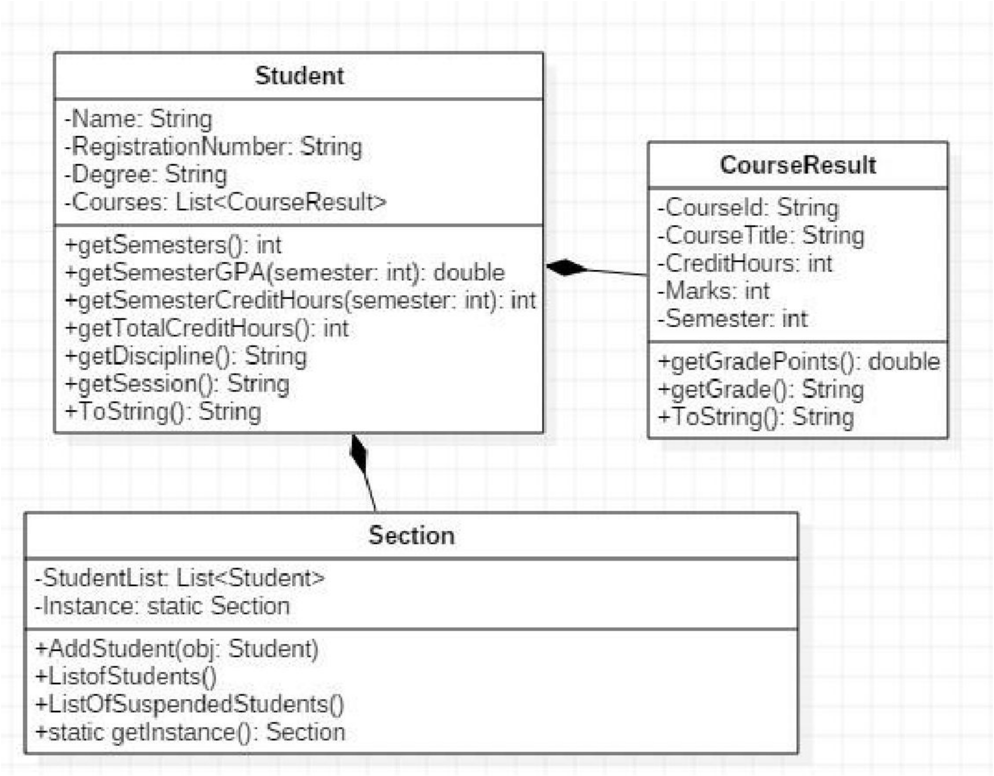

Student
-Name: String
-RegistrationNumber: String
-Degree: String
-Courses: List<CourseResult>
+getSemesters(): int
+getSemesterGPA(semester: int): double
+getSemesterCreditHours(semester: int): int
+getTotalCreditHours(): int
+getDiscipline(): String
+getSession(): String
+ToString(): String
CourseResult
-CourseId: String
-CourseTitle: String
-CreditHours: int
-Marks: int
-Semester: int
+getGradePoints(): double
+getGrade(): String
+ToString(): String
Section
-StudentList: List<Student>
-Instance: static Section
+AddStudent(obj: Student)
+ListofStudents()
+ListOfSuspendedStudents()
+static getInstance(): Section

Figure 1 Class diagram

- **StudentName** //should be alphabetic, special characters and numbers are not allowed
- **RegistrationNumber** //Format should be like this: 2015-CS-888, any other format should be handled in setter function
- **Degree** //it should be MS, BS or BE
- **CourseID** // Format should be valid according to your course codes given in your LMS. For instance, a software engineering lab has a course ID of CS381L. Length of course code should be from 2 to 8 characters.
- **CourseTitle** // should be alphabetic. Length of course code should be from 10 to 35 characters.
- **CreditHours** // values from 1 to 3 are allowed

- **Marks** // values from 0 to 100 are allowed
- **Semester** // valid range is from 1 to 8

1. Your Program should define three constructors for class of **CourseResult**
    a. a constructor with **no parameter**
    b. a constructor with **parameters**
    c. **copy** constructor
2. **Student** will have only one constructor without parameter
3. Define getter setter for each data member in classes
4. Apart from getter, setter and constructor, define the following functions in respective classes according to class diagram
    a. **getGrade()** – it should calculate grade based on marks using the following criteria.
        - IF marks are less than 40 – Grade is F
        - IF marks are between 40 and 50(exclusive) - Grade is D
        - IF marks are between 50 and 55(exclusive) - Grade is C
        - IF marks are between 55 and 60(exclusive) - Grade is C+
        - IF marks are between 60 and 65(exclusive) - Grade is B-
        - IF marks are between 65 and 70(exclusive) - Grade is B+
        - IF marks are between 70 and 80(exclusive) - Grade is A-
        - IF marks are above 80 - Grade is A
5. **getGradePoints()** – function should return grade points using the following criteria

Grade	**CoursePoints**
A	4.0
A-	3.7
B+	3.3
B-	3.0
C+	2.7
C	2.3
D	1.0
F	0

i. **getSemesters()** – it should return number of semesters based on course list

getSemesterGPA(semester: int) – calculate semester GPA according to following formula

$$SemesterGPA = \frac{\Sigma(SemesterCourseGradePoints*CreditHours)}{SemesterCreditHours}$$

getCGPA(): calculate GPA using the following formula $CGPA = \frac{\Sigma(CourseGradePoints*CreditHours)}{TotalCreditHours}$

ii. **getTotalCreditHours()** – it should return number of credit hours based on course list

iii. **getSemesterCreditHours(semester: int)** – it should return number of credit hours for a given semester based on course list

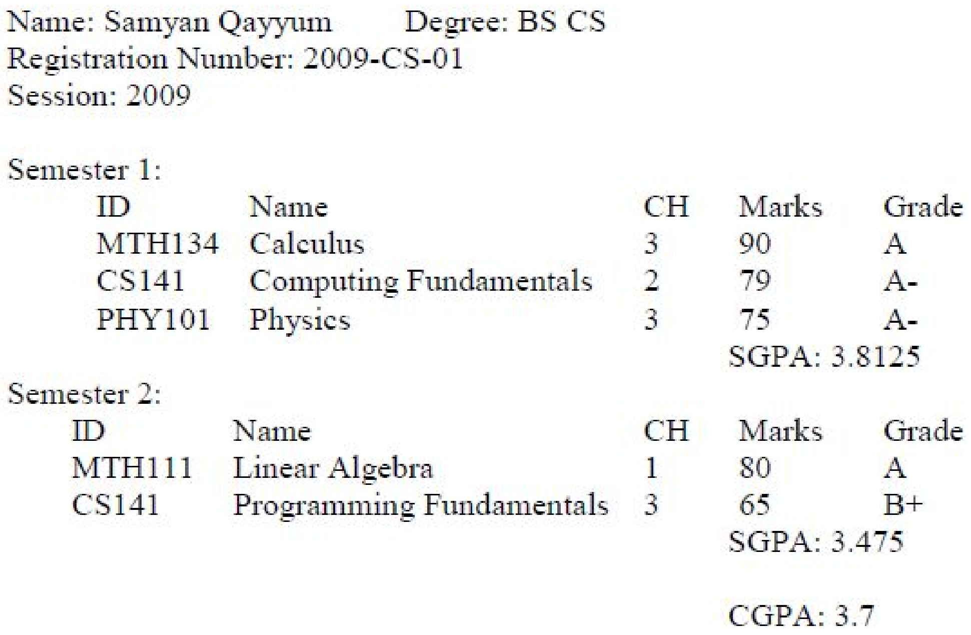
iv. **getSession()** – extract session from RegistrationNumber

v. **getDiscipline()**: extract discipline from RegistrationNumber

vi. **toString()** – Purpose of this function is to write all attributes of a class in desired format and return as a string

vii. **ListOfSuspendedStudents():** this will return students with CGPA less than 2.0.

- Create GUI to support the above class diagram. GUI will be your choice, better GUI will get more points
- You don’t need to write data in file or database.
- Use DataGridView for Course List and Student List
- ToString() of Student will show result in following format :



```
Name: Samyan Qayyum        Degree: BS CS
Registration Number: 2009-CS-01
Session: 2009

Semester 1:
        ID          Name                        CH    Marks    Grade
        MTH134      Calculus                    3     90       A
        CS141       Computing Fundamentals      2     79       A-
        PHY101      Physics                     3     75       A-
                                                      SGPA: 3.8125
Semester 2:
      ID          Name                          CH    Marks    Grade
      MTH111      Linear Algebra                1     80       A
      CS141       Programming Fundamentals      3     65       B+
                                                      SGPA: 3.475

                                                      CGPA: 3.7
```

Figure 2: Output format