Colorado Mesa University

Computer Science & Engineering

CSCI112 **Data Structures** **Spring 2021**

Assignment 1-P1

This part is Due: 11:00 pm Monday 22 February, 2021 **20 points**

Aim:

This assignment is part 1 of 2 and aims to familiarise you with the use of classes in your programs. **On completion you should know how to:**

- Write programs with class objects.
- Implement programs incrementally to minimise debugging.
- Produce a database program for maintaining student records.

Requirements:

You are to complete a program that is designed to access University enrolment records. The program is to offer the following options:

- List all subjects offered in a semester.
- List subjects that a student has taken.
- List all students enrolled in a subject.
- Find students that have less than n credit points.
- Quit

A typical test run of this program is shown below:

```
*** University Enrolment System ***

1. List all subjects offered in a particular semester
2. List the subjects that a particular student has taken
3. List all students enrolled in a particular subject
4. List the students that have less than n credit points
5. Exit from the system
Enter your choice (1..5): <1>

Enter:
0 : for annual subject
1 : for first semester
2 : for second semester
Which semester? <1>

Subjects offered are:
113     Fluid Mechanics A
115     Networks A
201     Chemistry 1
121     Introduction 1


*** University Enrolment System ***

1. List all subjects offered in a particular semester
2. List the subjects that a particular student has taken
3. List all students enrolled in a particular subject
4. List the students that have less than n credit points
5. Exit from the system
Enter your choice (1..5): <2>

Enter student's ID : <90000>

This student does not exist!

Enter student's ID : <123456>

Student ID        : 123456
Student Name      : Duffy Duck
Enrolled subjects : 111 121 131
```

```
*** University Enrolment System ***

1. List all subjects offered in a particular semester
2. List the subjects that a particular student has taken
3. List all students enrolled in a particular subject
4. List the students that have less than n credit points
5. Exit from the system
Enter your choice (1..5): <3>

Which subject code do you want to display? <101>

Subject cannot be found!

Which subject code do you want to display? <111>

Subject Code : 111
Subject Name : Mathematics 1A

Enrolled Students
-----------------
Duffy Duck
Donald Duck
Winnie The Pooh
Tigger
Hunny Bear


*** University Enrolment System ***

1. List all subjects offered in a particular semester
2. List the subjects that a particular student has taken
3. List all students enrolled in a particular subject
4. List the students that have less than n credit points
5. Exit from the system
Enter your choice (1..5): <4>

Enter credit point limit? <10>

Students who have taken less than 10 credit points:

Duffy Duck                    4
Donald Duck                   5
Mickey Mouse The Rat          8
Danoz Direct                  2
Minnie Mouse                  5
Optiplex G Dell               8
Optima Good                   5
Pigeon Hole                   9
Felix The Cat                 9
------------------------------------------------------------
There are 9 students in this category.


*** University Enrolment System ***

1. List all subjects offered in a particular semester
2. List the subjects that a particular student has taken
3. List all students enrolled in a particular subject
4. List the students that have less than n credit points
5. Exit from the system
Enter your choice (1..5): <5>

Thank you for using University Enrolment System.
```

Implementation:

A partially completed implementation of this program is provided in the following files:

main.cpp	text menu based user interface
Student.cpp, Student.h	class object for storing each student's details
StudList.cpp, StudList.h	class object for containing a number of Student objects
Subject.cpp, Subject.h	class object for storing each subject's details
SubjList.cpp, SubjList.h	class object for containing a number of Subject objects

Before writing any code in any of the above files create a project and add the above files to it. Now compile, link and run the program to ensure there are no errors and the menu gets displayed ok. As you complete each step (below) recompile and run your program to test that each step is correctly implemented. Briefly browse through each of the above files to see how the 4 classes are implemented and get some idea of how much more work is required to complete each class.

Step 1 requires you to implement the constructors, destructor and the ReadFile() public member functions of the SubjectList and StudentList classes. To do this, you will also have to implement the constructors of the Student and Subject classes. When the program begins the main() function creates an instance of a StudentList and a SubjectList. The StudentList and the SubjectList are then instructed (via their ReadFile() public member functions) to read the data stored in *student.txt* and *subject.txt* respectively. As the data for each student or subject is read, a new instance of a Student or Subject class is created for storing the data. (Note: the initialisation constructors of the Student and Subject classes should enable each instance to be passed its data upon creation.) The address of the newly created objects are stored in the array provided in each class's private member section. First perform this step on the SubjectList class and then test your code by compiling and running your program and selecting option 1 of the menu. Then do the same for the StudentList class in preparation for Step 2.

Note: The *StudentList* and *SubjectList* data are stored in *text files* named *student.txt* and *subject.txt* respectively.

The data for a student consists of 3 lines:

- ID number which is a character string of length 6,
- Name which is a character string of length 25,
- The number of subject taken by the student, followed by a list of subject Ids (up to 7).

The data for a subject consists of 4 lines:

- Subject code which is a character string of length 3,
- Subject name which is a character string of length 25,
- Credit points which is an integer, and
- A *session flag* (integer) that is
  - 0 if the subject is annual
  - 1 for first semester
  - 2 for second semester

Step 2 For step 2 you should provide the functions required for menu option 2. This requires you to implement the public member function named PrintStudentDetails() of the StudentList class. You will also have to declare and define some public member functions of the Student class for accessing and printing this class's private data. A possible algorithm for the PrintStudentDetails() function is shown below. (Note: make sure the output of all functions complies with the sample program run shown on page 1.)

```
int StudentList::PrintStudentDetails(char StudentID[])
{
        for each student in student array{
                if ith array's student ID is same as StudentID then{
                        print student ID
                        print Student Name
                        print Enrolled subjects
                        return 1
                }
        }
        return 0
}
```

Step 3 Implement menu option 3 by following a similar procedure to what was described in Step 2. This requires you to implement the PrintSubjectDetails() and PrintStudentsInSubject() public member functions of the SubjectList object and the StudentList object respectively. An example of what these functions should do when the test data is loaded is shown in the sample run on page 2. For example, when PrintSubjectDetails() is called with "111" as its parameter it should find the 111 subject in the subjects array and print the following on the screen:

```
Subject Code : 111
Subject Name : Mathematics 1A
```

Likewise, when PrintStudentsInSubject() is called with "111" as its parameter it should display on the screen:

```
Enrolled Students
-----------------
Duffy Duck
Donald Duck
Winnie The Pooh
Tigger
Hunny Bear
```

Step 4 Implement menu option 4 by following a similar procedure to what you did in Step 2 and 3. This requires you to implement the StudentList public member function: PrintStudentsBasedOnCredit(). This function is a little different to those described above in that it is passed the SubjectList instance to enable the Student class to access the names and credit points of subjects from the subject codes stored in the Student class. A possible algorithm for implementating this function is shown below. Implementation of the required data access functions of the Student class is left up to you.

```
void StudentList::PrintStudentsBasedOnCredit(int CPLimit,SubjectList &Subjects)
{
        set student counter to 0;
        for each student in student array{
                instruct ith Student in array to->GetCreditPoints(Subjects);
                if CreditPoints < CPLimit then{
                        print ith Students name and credit points
                        increment student counter;
                }
        }
        print number of  students in this category
}
```

Step 5: Throughly test your program to ensure its output complies with the example output shown above. Also make sure all files have the same names and case as specified in the submission instructions. Points will be deducted for failing to strictly comply with the code layout and style instructions from CS1, the use of blacklisted libraries shall result in 0 grade award (all STL are blacklisted).

Submit:

You have to submit the following files on on D2L in a single compressed archive, test compile error and warning free on Ubuntu using C++ 14: ***main.cpp, Student.h***, ***Student.cpp, StudList.h***, ***StudList.cpp***, ***Subject.h***, ***Subject.cpp***, ***SubjList.h*** and ***SubjList.cpp.***

I will assign you a demo / run time where we discuss your assignment submission. The assignment deadline will be strictly enforced. Assignments will not be accepted if more than 1 day late (50% penalty for late submission). Demo & discourse is the only eval option. Part 2 will have its own unique deadline.