INB371 Assignment 1

Student Assessment Reporting System

Due Date: 2 April 2010

Weighting: 20%

1. Preamble

The end of the semester is approaching. You are hired by the Student Services Department in a University to develop a C++ program that can produce students' assessment result reports for this semester as soon as the students' assessment information is available.

For this assignment, we assume that students fall into two categories: course work students and research students. For all students, the basic information about a student includes the student's name (first name and last name), the student's ID, and a category that indicates which category the student belongs to. For course work students, the specific information includes the number of units the student has taken in this semester and the grades that the student has achieved for each unit. For research students, the specific information includes the degree type (PhD or Master) and a study status (newly-enrolled, topic-confirmed, candidature-confirmed, or degree-completion). We assume that the students' IDs are unique and each student can be either a course work student or a research student, but not both.

Your program should be able to read student information from keyboard, store the data, compute final grades for course work students or determine the next milestone to achieve for research students, and display a result report onto the screen.

2. The Task

1) Design and implement three classes that store and process student information. There should be **one base class** to model general student objects, and **two derived classes** to model course work student objects and research student objects, respectively. All data items declared in these classes should be private or protected. Any access to class data from outside should be done through public member functions.

   (a) Base class for general students – class ***student***

      i. Including data items for student first name, last name, ID number, and category

      ii. Including the following operations:

         - Constructors
         - Accessor and mutator functions if necessary
         - Display the student information

   (b) Derived class for course work students – class ***courseStudent***

      i. Including data items for the number of units taken, the unit code, and the grades for each unit (assume that the grade scale is 1 – 7.)

      ii. Including the following operations:

- Constructors
- Accessor and mutator functions if necessary
- Calculate the student's average grade (GPA) for the units taken in this semester
- Display the student information

  This operation should reuse the "display the student information" operation in the base class to display the student's basic information, and then display the units, the corresponding grades, and GPA.

(c) Derived class for research students – class ***researchStudent***

i. Including data items for the degree type undertaken and the study status of the student.

ii. Including the following operations:

- Constructors
- Accessor and mutator functions if necessary
- Calculate the student's next milestone. The next milestone for a newly-enrolled, topic-confirmed, candidature-confirmed, and degree-completion student should be proposal-submission, confirmation-report-submission, thesis-submission, and completion, respectively.
- Display the student information

  This operation should reuse the "display the student information" operation in the base class to display the student's basic information, and then display the student's degree, currently achieved milestone, and the next milestone that the student needs to achieve.

2) Write a client program that reads student data from the keyboard and stores it using an array of appropriate type. You should use just **one array** for all students (i.e. this will be a **heterogeneous list**). You don't have to use a dynamic array. You can define an integer constant (a value big enough to hold all students) to be the size of the array. Then declare an array with this size.

Your client program should repeatedly allow the user to select the following operations until the user chooses 'Exit':

1. Add a course work student into the student array.
2. Add a research student into the student array.
3. Output student assessment results on the screen for course work students only. The output should include students' basic information, units, grades, and GPA.
4. Output student assessment results on the screen for research students only. The output should include students' basic information, degree, the milestone achieved and the next milestone.
5. Output student information and assessment results for all students.

6. Exit.

3) File input (optional)

You can input students' information from a file. This is an optional operation.
A sample input file may look like the following:

```
6
Bugs Bunny 12345678 course
4 ITB111 6 ITB222 7 ITB123 5 ITB720 6
Joe Schmuckatelli 11111111 course
3 ITN700 3 ITN702 5 ITN400 4
Marvin Dipwart 22222222 research
master newly-enrolled
Jimmy Crack-Corn 33333333 research
phd candidature-confirmed
James Kirk 44444444 course
4 INB371 5 INB310 6 INB230 6 INB111 6
Monica Lewinsky 55555555 research
phd degree-completion
```

The first line of the above input file is the number of students listed in the file. After the first line, every set of two lines constitutes a student entry.
The first line of a student entry contains the name, ID number and category. For course work students, the second line contains the number of units taken and a list of unit-code and grade pairs. For research students, the second line contains the degree type and the status of the student.

3. General Requirements

- Your program must be a console application implemented in C++ and works in Microsoft Visual Studio .NET.
- All member data of your classes must be private or protected
- Any dynamically allocated space should be cleaned up appropriately with `delete` when you are finished with it.
- Submissions without any code will be heavily penalised. Submissions that partially work will be marked much less harshly.

4. Submissions

Your assignment solution should be submitted via the Online Assignment Submission (OAS) system before 23:59pm, 2 April 2010. Your submission should be a single zip file named by **report.zip**, and comprises of the complete Microsoft Visual Studio 2008 Win32 console application project, and a single Microsoft Word document containing the following:

- Table of contents
- Statement of completeness – it specifies if the assignment has been fully completed and if the system is fully functioning. If the assignment has not been fully completed or the system does not fully function, you should specify them in the statement.
- Test plan and test results for each of the required operations – it should be a tabular format indicating what was being tested, what the purpose of the test was, what was done to test it, what the expected result was, and whether the test was passed.

5. Marking Criteria

Class declaration and implementation (50%)

Base class declaration
Derived classes declaration
Function prototypes, compiler directives, access levels, pre-conditions, post-conditions
Class implementation, variables (data structures)
Class implementation, functions

Client program (30%)

Data structures, memory deallocation
Functionalities

Testing (15%)

Test plans are well designed (the test cases, actual input, and expected output should be provided)
Test results are provided

Presentation (5%)

The assignment is neatly presented
The statement of completeness is included
Table of contents is included