CS 1410 Introduction to Computer Science – CS2
Assignment #7: OOP
Total Points: 50 points

**Problem description**

Your task is to design and implement a management system for a university. In this assignment, you will design and declare the class hierarchy of this system. A class (e.g. Person) in this system is declared in two files. The header file (e.g. Person.h) contains the definitions for the fields, their getters/setters, the constructors, and the prototypes of other methods. The source code file (e.g. Person.cpp) contains the implementations of those methods. You should compress all those files and submit as a file named as Firstname_Lastname_HW7.zip

**Programming tasks**

**Task 1 (20 points)**. Declare the following classes.

a. Class Course has course ID, name, number of credits, grade, semester (e.g. “Spring”, “Fall”, “Summer”), and year (e.g. 2014). Grade can be A,A+,A-,B,B+,B-,C,C+,C-,D,D+,D-,F,I (incomplete), and X (withdrawn). An example of a course is [ID = “CS1410Sp2015”, name = "Introduction to Computer Science 2", credit = 3, grade = "A", semester = "Spring", year = "2015"]. You need to implement getters and setters for all those fields and a default constructor that assigns empty value for all those fields (e.g. number of credit = 0, grade = ""...).

b. Class Person has ID, name, date of birth, gender, and address. You need to implement getters and setters for all those fields. The default constructor should assign empty value (e.g. "") for all those fields.

c. Class Professor is a subclass of Person. The new data includes job title, office, and salary. You need to implement getters and setters for them.

d. Class Student is a subclass of Person. The new data includes the major and the list of courses. The list of courses is declared as a vector of pointers to Course objects, i.e. vector<Course*>. You need to implement the getter and setter for major. The default constructor should assign the major as an empty string and the list of course as an empty vector.

**Task 2 (8 points)**. Implement two methods 'read' and 'write' for all aforementioned classes. Method 'read' gets all information for the simple fields (i.e. not the fields declared as vectors) from cin, while method 'write' prints all fields to cout. As different classes will perform read/write differently, you should implement them as **virtual functions** and call the read/write method of parent classes when needed.

**Task 3 (7 points)**. Add the following functions to class Student:

a (3 points). A method to compute GPA from the list of courses. A letter grade is converted to a numeric value using typical rules, i.e. A/A+ = 4, A- = 3.67, B+ = 3.33, B = 3... Incomplete and withdrawn courses are not counted towards GPA.

b (3 points). A method to print the transcript with all available information (e.g. name, gender, major, GPA, courses...).

c (1 point). A method to add a course for a student (i.e. in its vector<Course*>), given the pointer to that course. Its prototype is:

void addCourse(Course *course);

**Task 4 (15 points)**. Write your main program with the following requirements:

a. Have a list of Person objects (as a vector of Person pointers, i.e. vector<Person*>).

b (1 point). Have a function to search for a person in the Person* vector given his/her ID. Its prototype is

Person* findPerson(string ID)

c (3 points). Have a function to add a new person (who could be a professor or a student) into that list. This function should have a parameter for the type of person the user want to add (e.g. a professor). It creates the corresponding object on the heap memory (i.e. via a pointer and the 'new' operator), use its read method to read the needed information, and then add the pointer to the Person* vector.

d (3 points). Have a function to add a new course for a student. You will use method findPerson to look for that student and get a pointer (Person*) points to the corresponding Student object. Then you create a Course object on the heap and read its information. Finally, you cast the Person* pointer to Student* pointer and call the method addCourse (Task 3c) with the Course* pointer.

e (8 points). Have a menu that lists all the available functionality (e.g. add a person, add a course for a student, print transcripts...). When the user requests a menu item, your program should perform the corresponding function.

Sample input and output (text within brackets [ ] is user input).

The menu:

Select your choice:
1. Add a professor
2. Add a student
3. Add a course for a student
4. Print the transcript of a student
5. Print information of a professor
6. Quit

<u>Usage when the user chooses item 2</u>:

ID: [1234]
Name: [Adam Smith]
Date of Birth: [10/10/1990]
Gender: [Male]
Address: [Logan, UT]
Major: [Computer Science]

Usage when the user chooses item 3:

What is the student ID? [1234]
Course ID: [CS1410Sp2015]
Course name: [Introduction to Computer Science 2]
Semester: [Spring]
Year: [2015]
Number of credits: [3]
Final grade: [A]

Usage when the user chooses item 4:

What is the student ID? [1234]

STUDENT TRANSCRIPT

Student ID: 1234
Name: Adam Smith
Date of Birth: 10/10/1990
Gender: Male
Address: Logan, UT
Major: Computer Science

List of courses:

Course ID: CS1410Sp2015
Course name: Introduction to Computer Science 2
Time taken: Spring 2015
Number of credits: 3
Final grade: A

Course ID: CS1400Fa2014
Course name: Introduction to Computer Science
Time taken: Fall 2014
Number of credits: 3
Final grade: A-

Course ID: MATH1100Fa2014
Course name: Calculus 1
Time taken: Fall 2014
Number of credits: 4
Final grade: B

Course ID: MATH2100Sp2015
Course name: Calculus 2
Time taken: Spring 2015
Number of credits: 3
Final grade: X

Cummulative hours: 13. Cummulative GPA: 2.69