CS 201R
Summer 2016
Program 4 – Dynamic memory in classes

Wassamatta U. is updating its online registration system; you've been assigned writing part of the course-enrollment section.

A Student has a first and last name, and ID number, and a variable number of courses. The number of courses a student has completed can range from 0 (for the just-accepted incoming freshman who has taken no coursework yet) to a very large number (for the eighth-year perpetual student who has changed majors six times). Thus there is no 'standard' size, and it makes sense to use a dynamic structure within each Student object to hold the data.

You will need a Student class. A Student has the following public methods:

- Because it uses dynamic memory, it will need a default constructor, copy constructor, destructor, and overloaded assignment operator.
- Getters and setters for: ID number (integer), first and last name (strings).
- bool AddCourse(const Course&) and bool RemoveCourse(const Course&). See below for definition of the Course struct. These add and remove, respectively, courses from the student's dynamic list, returning true if the operation was successful, false otherwise.
- bool IsEnrolled(const Course& aCourse) const. This method takes a const reference to a Course and and returns true if the student is enrolled in the course, false otherwise.
- bool ReadData(istream& in). This method reads a student's ID number, first and last name; an integer specifying how many courses they are enrolled in; then the correct number of Courses. It returns true if all attempts to read the data and add it to internal data structures were successful.
- bool WriteData(ostream& out). This method writes the student's data in the same format the ReadData() method expects: the ID number and first and last name, all on one line, separated by whitespace, then the number of courses, on a line by itself, then each course on a line by itself.
- double TuitionDue() const. This method computes the student's tuition bill on a credit-hour basis using rules defined below.

You will need a Course struct. It will have:

- The course department (string), number (integer), and credit hours (integer).
- Overload the stream insertion and extraction (<< and >>) operators for this struct.

Tuition computation:

- Courses cost $476 per credit hour.
- Courses in computer science, information technology, or any engineering course (department CS, IT, ECE, CE, ME) have a $15 per credit hour surcharge to cover lab expenses.
- All students pay a flat $10 library fee, a $0.75 per credit hour student union fee, and a 0.25% charge for the Chancellor's Pastry Fund. (Watch your decimal points on that—that's one-fourth of one percent, computed after all other fees are added.)

Extra credit! For 5 points extra credit, overload the stream insertion and extraction operators for the Student class.

You are given two data files produced by the Wassamatta U. enrollment system. The first is a list of all students enrolling for courses. You will need to store these in a dynamic array in the main program.

The second is a series of entries of: Student ID, Course. For each case, add the course to the student's roster. You will then produce 2 output files:

The first lists the students in the following format:
ID number First name Last name
    course
    course
    course (etc., for however many courses the student is taking)
Tuition.
// blank line
ID number, etc. for next student.

The second file lists:
Course information
    Student_ID First name Last name
    Student_ID First name Last name
    and so on, for all students in the course
blank line between courses

You are given a set of small input files. Your program will be tested on a larger file with the same format.