CSCE 121(505-508) **Homework 10**
Due 11:59 p.m., 4/24/14 [175 points]

This Homework assignment is meant to help you understand more about containers, iterators, and algorithms, and to give you more experience with classes.

All work should be turned into CSNet by the deadline. **In addition, please turn in a hardcopy to the TA's mailbox within one day of submission.** For programs, you need to turn in only the source code (not object or executable code). Your code will be tested using g++: you are welcome to develop in Visual Studio, but please make sure your code also runs in g++.

-1) For going through the containers (such as vectors) in this assignment, you are to use iterators.

0) Create a text file, README, in which you:
   a. State the Aggie Honor statement, or else explain why you cannot do so.
   b. List any resources used, outside the textbook and discussions with the Instructor, TA, or Peer Teacher
   c. List any known problems with the assignments you are turning in. For example, if you know your code does not run correctly, state that. This does not need to be a long explanation.
   d. For places where you may have done some additional work, put in a brief summary detailing what you did.

1) [175 points] You are to define two classes, Student and Course, and perform several operations on them:
   a. [15 points] The Course class should include a department (such as "CSCE"), a course number (such as 121), and a grade (A, B, C, D, or F). You may include other data if you wish. Write a routine to output a course and be sure you can input one from some file (see part e and following).
   b. [15 points] The Student class should contain a student name and a vector of Courses the student has taken. You may include other data if you wish. Write a routine to output a student and be sure you can input one from some file (see part e and following).
   c. [40 points] Using the sort command and by defining the appropriate comparison functions, provide the ability to sort a student's courses by letter grade, or by department/number (i.e. alphabetically by department, then by number within that department).
   d. [30 points] Write a routine that computes the GPA for a particular student. You should use iterators to handle the list of courses and use the accumulate algorithm as a key part of the calculation. Note that you will need to write the binary operation function to pass in to accumulate.
   e. Create a file with at least 10 students' data. You can define this format however you want. Your students should not already be in alphabetical or grade-point order, and the courses should not all be in order by grade or by department/number
   f. [20 points] Read in the file to create a list (not vector) of Students. Sort the students by name and output to a file. For each student, sort the courses by department/number.
   g. [20 points] Read in the file to create a vector (not list) of Students. Sort the students by GPA and output the students to a file. For each student, sort the courses by letter grade.
   h. [35 points] Read in the file. Using a map (a different but similar container is OK to use), output a list of all the courses taken by all the students, and the number of times that course has been taken. For example, if there were 2 students, and one of them has taken CSCE 121 and CSCE 221, and the other has taken MATH 151 and CSCE 121, you would output something like:

      CSCE 121: 2
      CSCE 221: 1
      MATH 151: 1

      The order the courses are printed out does not matter.
      hint: see section 21.6 where words are counted. You will want to make a department/number structure that can be used as the map index instead of just the string in the example.
   i. [40 points EXTRA CREDIT] Output the result of h in order:
      i. given by the department/number
      ii. given by the from most frequent to least frequent.
   j. [35 points EXTRA CREDIT] Create another data file with at least 3 students identical to those in the first file, and at least 3 more different from those in the first file. Using the merge() algorithm, read in the two files and merge them into one, removing duplicates.