Santa Monica College
CS 20A, Fall 2014

Programming Project 1 - Due: Sunday, September 21 at 11:59 PM

You are a programming intern at the alumni relations office. Having heard you are taking CS 20A, your boss has come to ask for your help with a task.

Quite often she needs to process files containing records for students attending UC schools and find those who are SMC alumni. She needs you to write a program that takes in a file containing UC student records and another file containing SMC alumni records and writes out to a file all the SMC alumni who are attending UC schools, ordered by ID.

This sounds like a very simple task and you immediately agree to work on it. She warns you that the files are pretty large - a few hundred thousand student records. You inform her that with today's computers having processors of a few GHz this task is a piece of cake and can be computed in an instant.

Part 1:

You think about the problem for a minute and you come up with a plan.

- You will create a Student class representing student records. It will have three data members corresponding to the data contained in the student records file

```
int id;
string name;
string school;
```

- You will implement operator = = to consider two student objects as equal if they have the same ID.
- You will implement operator < to allow sorting of student objects by ID.
- You will implement the toString method to print the fields as they appear in the student records file.

The idea is that for every record you read from the file, you will create a Student object. You will store the UC students into a student array and the SMC alumni into another student array. Parts 2 and 3 operate on these arrays.

Part 2:

Now that you have two arrays with one containing UC students and the other containing SMC alumni, you plan to process the arrays and find the students who belong to both arrays. Your data structures are these two arrays. Your algorithm will be: for each student in the UC students array you try to find it in the SMC students array by comparing it for equality with every student in the SMC students array. If you do find it, that UC student is then stored in an array containing students who are SMC alumni attending UC schools. Once you have the array of the common students, you sort it by ID and print it to a file. We will use the C++ Standard Template Library sort for this (already done for you in main()).

Part 3:

After running your program from Part 2 and discovering that it takes about a minute or more to complete for the given input (much longer for larger files), you decide to really think about the problem in order to find a solution that does not force your boss to go get a cup of coffee every time she needs to run your program. It seems even those powerful processors in today's computers can't really save a bad algorithm.

You remember from class that binary search is much more efficient than linear search for finding an element in an array and decide to try it out. We will use the C++ Standard Template Library binary search for this. Your algorithm will be: for each student in the UC array you will binary search the sorted SMC array. If the binary search finds the student, that UC student is stored in an array containing students who are SMC alumni attending UC schools. Once you have the array of the common students, you sort it by ID and print it to a file.

Hints:

It is always a good idea to test each little piece after you complete it. Trying to debug a whole program is hard and very time consuming.

Whenever you work with a large data set it is much easier to first work on a small data set to ensure your program functions correctly. Once you are confident that it produces the correct result you can move to the larger data sets. To help get you started, two small sample input files and the expected output have been provided.

Deliverables:

Since this is the first project, the design and most of the work has been done for you. You should implement the Student.h header and fill out the missing code in the provided program.

You should submit a zip file named project1_first_last.zip (where first and last are your first and last name) containing **ONLY the files below.**

Student.cpp	
Project1.cpp	
output.txt	// a copy of the program output (copy and paste it into this file) containing // ONLY the timing printouts
smc_grads_at_uc_2.txt	// or .._1 if you do not get _2 working

How you get points:

– Student.cpp	
constructors	8 points
implementing operator = =	8 points
implementing operator <	8 points
implementing toString function	8 points
– Project1.cpp	
readStudentsFromFile	13 points
writeStudentsToFile	5 points
findCommonStudents1	25 points
findCommonStudents2	25 points

How you lose points:

- You do not follow the given directions and decide to make changes "for fun". Specifically, **do not change the skeleton code given to you**. Just fill out the missing parts.
- You use Vector or any data structure we have not yet seen in this class. **Only arrays are allowed.**
- **Submit only the files the project asks for.** Not your workspace or executable files.
- Your solution should be **efficient to a level seen in class for similar problems**.
- If any of your code prints anything at all on the console except for the timing output. **Remove all your print outs, debug statements, etc. Clean up your code and do not leave clutter behind.**
- **Comment your code appropriately.**