Programming Assignment #4
CS 162: Introduction to Computer Science

Submit your assignment to the D2L Dropbox
Email a backup copy to karlaf@pdx.edu

The purpose of the 4th program is to implement the new concepts learned which include (a) pointers, (b) dynamically allocated arrays, and (c) classes. Our goal is to continue to create programs with a small main function that delegates to a series of functions where the real work takes place. Place your class interface in a .h file and the class implementation in a .cpp file.

In this programming assignment, you are **not** allowed to use global variables. You are allowed (as usual) to use the cstring library (e.g., strlen, strcpy, and strcmp). Limit your main (and all functions) to no more than 30 statements of code (for executable statements… *not counting variable definitions, blank lines, lines with just curly brackets, or comments*).

Program Assignment:
Keeping track of attendance is almost a full time job for one of the lab administrators. Some of the professors use clickers instead to avoid needing to take attendance. But even that information needs to be merged with a master attendance list which is no easy task. Writing a program that would take your attendance entry and automatically place it in the master list would be very helpful!

Your job for this 4th assignment is to build a **class** called "Student" that will keep track of a single student's attendance records. It will keep the lab attendance separate from the lecture attendance. It will need a dynamically allocated array for the lectures and labs that are attended. With dynamically allocated arrays, you will also need two integers to keep track of how (a) full the array is and (b) the maximum size of the array. Do not use a constant for the size of these arrays!

Once this is working, build a class called "Attendance" that will keep track of all of the attendance for all students in a given class. You will need to use a dynamically allocated array of students and a dynamically allocated array of characters for the course name.

The following is a suggested set of classes. You may add more or modify the arguments or functions:

```
class Student
{
    public:
                Student();
                ~Student();  //A destructor used to deallocate dynamic memory
                void Attend_Lab(int lab_number);
                void Attend_Lecture(int lecture_number);

                //Only displays the information if the student name matches
                void Display(char student_name[]);

                //Displays the attendance information, including the name
                void Display();
  private:
    /*Put a dynamically allocated array for the student's name, a dynamically allocated array for the number of labs in a term, and a dynamically allocated array of the lectures in a term; You will also need integers for the number of labs being attended and the number of lectures attended as well as for the sizes of those arrays */
};

class Attendance
{
    public:
                Attendance();
                ~Attendance();  //A destructor used to deallocate dynamic memory
                void Read_Attenance(); //Received weekly attendance information
                void Display_Attenance(char student_name[]);

                void Display_all();
  private:
    /*Put a dynamically allocated array of students in the class and an integer count of the number of students */
};
```

Extra Credit:
Try out constructor with arguments:

```
                Student(int num_lectures, int num_labs);
                Attendance(int num_students);
```

***You are always welcome to do more! But, really focus on making general purpose functions that can be re-used. Anytime you have code that has already existed elsewhere in your program (such as to error check input or give the user another chance), write a function instead!*

In Program #5 we will store the students in a linear linked list!

Things you should know...as part of your program:

1. Make sure to prompt the user for any input requested. Make sure it is clear from your prompts what the user is expected to do.
2. Have main test out all of the functions to make sure they work for all conditions.
3. The program should continue until the user wants to quit. Allow them to continue until they are done.
4. Make sure to put your name in your program
5. Submit an electronic copy of your .cpp file as an attached file to the dropbox on D2L (go to: http://d2l.pdx.edu/ to login). Make sure to hit the submit button after uploading your files (otherwise they will be lost)