CSCE 121(505-508) **Homework 6**
Due 11:59 p.m., 3/18/14 [150 points]

This Homework assignment is meant to (a) help you get used to using the graphics commands that will be used in the next part of the course, (b) give you experience with working with object-oriented calls, and (c) give you practice with some concepts that we discussed but did not use much previously.

All work should be turned into CSNet by the deadline. **In addition, please bring a hardcopy printout of your code to class the next day.** 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++.

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) [150 points] You are to create a chessboard that could be used to draw the state of a chess game at any point in time.
   - This will have several parts to it, and you should generally work on these in order. Don't go on to the later parts until you have the earlier ones working. Make sure to test each part before going on to the next one.
   - Note: part of the goal here is to use "good" program design to accomplish the following. Think about how you can do each of these things in a way that is "elegant", rather than brute-force. Generally, an elegant solution will be described via a standalone function call or something similar, while a brute-force approach will hardcode in specific values one-by-one.
   - If you are not familiar with a Chess board setup, you can find a simple diagram of the chess board (and links showing how to set the pieces on it for a real game) at: http://chess.about.com/od/rulesofchess/ss/Boardsetup.htm

   a. [25 points] You should create a checkerboard pattern on the screen. At minimum, draw grid lines separating the spaces. For full credit, the squares should be filled in. The typical colors of spaces are some variation of dark and light, though there are lots of variations (red/black, black/white, gray levels, etc.)
   b. [15 points] On the screen, label the different rows and columns (see the link above if you don't know the numbering)
   c. [10 points] Create an enumerated type for the two "sides" and another enumerated type for the different pieces.
      i. There are two sides, black and white
      ii. There are 6 different pieces: King, Queen, Bishop, Knight, Rook, and Pawn.
   d. [45 points] Find a way to display a particular piece within a square (e.g. square a1). You should use a different symbol for each of these pieces:
      i. For the Queen, draw a large circle
      ii. For the Bishop, draw an ellipse
      iii. For the Rook, draw a square
      iv. For the pawn, draw a triangle
      v. For the King, draw a "plus" sign (+)
      vi. For the Knight, find a picture of a horse image and draw it. You will need two different images, one for white and one for black.
   e. [15 points] Extend the previous routine to offset the piece to any of the squares on the board. You want to have a function that takes in a position and the piece, and draws that piece at that position. As one example, you might be able to have a command like: DrawPiece('c',3,black,pawn);
   f. [40 points] Create a Chessboard class that can keep track of which pieces are where
      i. Create a routine to let people "initialize" the board (i.e. clear it off entirely)

ii. Allow people to add a piece to the board – i.e. specify a piece and a position (like in part g). Keep track of all of the pieces and their locations.

iii. Allow someone to draw the current board. When the function is invoked, the basic chessboard should be drawn and all the pieces on the board should be displayed.

iv. For full credit, you should incorporate all the relevant previously defined functions into the Chessboard class. Think about what should be made public vs. private in an actual Chessboard class.

g. EXTRA CREDIT [50 points]. Using the above, create a program that lets a user play chess by specifying each move (you can get partial credit, the more of these you implement). Note: if you are not familiar with chess, you will have to look up basic rules.

i. Set up the board with the correct configuration.

ii. Allow a user to specify moves by naming a starting square and an ending square.

iii. Remove any "captured" piece (one that another piece is moved on to).

iv. Determine when someone has won (i.e. the King is captured).

v. Check that each move is valid. That is:

1. Pieces follow the rules of movement
2. Pieces don't land on another piece of the same color
3. Pieces don't illegally pass through other pieces
4. The correct player is moving a piece at each turn