CS2401 Project 6 Spring 2014
CHECKERS

We will devote most of the rest of the semester to the development of a text-based game of checkers in which the computer is able to play an "intelligent" game of checkers against a human opponent. To help us we will be **deriving** this game from the author's game class. You **must** use the author's game class which is described in section 14.3 of your text and the code for which can be found in the directory *~jdolan/cs2401/projects/game* . This will be a tightly integrated, multistage project, worth a total of 150 points, which means that as a whole it will represent approximately 14.6% of your overall grade in the course.

Design. Everyone's game will be derived the author's game class which is available in the directory mentioned above. Looking at this class you see that there are a lot virtual and purely virtual functions which means that most of the functions in the Checkers class that you write will have pre-determined names. Inside this Checkers class, the most important private variable will be a two-dimensional array of objects for the storage of the pieces/spaces of the gameboard. (Storing the board and displaying the board are two separate, but related things.) You will also be writing the class for these piece/space objects. It will need to store 1) whether or not a space is occupied, 2) the color of the piece occupying the space, and 3) whether or not that piece is a king. (Twenty years ago these could have been done with an integer codes, but this is object-oriented programming so we're going to use objects.)

Stage I: Development of the User Interface. At the end of this stage you should have created code that will display the board to the screen and allow the user to move one checker. Even at this beginning stage your class should be derived from the author's game class, but you are allowed to comment out most of the purely virtual functions, and I have altered the author's play function so that it only makes the one move. In your derived Checkers class you will need to implement your own versions of *restart( ), display_status( ), is_legal(const std::string& move),* and *make_move(const std::string& move).* The *play* function, that you are to call in your main, will also be calling *make_human_move*, and *get_user_move*, which you inherit, unaltered, from the parent class. Eventually, is_legal will encompass all the rules of the game, but for this stage it can simply check if the move is one of the 6 legal opening moves. This stage is worth 50 points and is due at 11:59 p.m. on Monday, April 14th.

~jedsubm/bin/2401submit -p 6

Note: At first it seems awkward to pass the moves around as strings. It may help to think of them as universal storage containers. Trying to fight this feature of the implementation leads to headaches down the road.

Stage II: A two player game. At this stage you will be implementing a two-player game of checkers. This means that you will have to implement the manner in which you are planning to accommodate the kings, and you will need to embody the rules of the game in *is_legal*( ), and probably extensively revise your *make_move( )* function. Remember that you can write helper functions for some tasks, including the business of being able to make multiple jumps. You will also need to implement the *is_game_over( )* function. When completed I should be able to play a two-player game of checkers with all the rules sufficiently enforced. This stage is worth 50 points and is due at 11:59 p.m. on Wednesday, April 23rd. ~jedsubm/bin/2401submit –p 7

Stage III: The computer plays. This is the part where you implement the "AI". A lot of this has already been designed in the author's game class so it is not as difficult as it might sound. In this stage you will need to implement *winning( ), clone( ), compute_moves(std::queue<std::string>& moves),* and *evaluate( )*. Most of your effort here will be in the coding of the last two functions. This stage is worth 50 points and is due at 11:59 p.m. on Sunday, April 28th. ~jedsubm/bin/2401submit –p 8

A summary of your schedule:

April 14th: User interface due. Game displays and plays one move.
April 23rd: Two player game due.
April 27th: Computer enabled game due.

P.S. Some rules and special things to note:

1. The forced jump rule: If a player has a jump available they must take the jump; if they have more than one jump available they may chose which one to take.
2. When a checker reaches the back row of the opponent's territory the checker becomes a "King" which gives it the ability to move in either direction.
3. Becoming a "King" marks the end of the turn. A player cannot make additional moves in that turn once becoming a king.
4. Since multiple jumps are possible there is a distinction between a "move" and a "turn," which can lead to confusion since the author is using the word "move" for both.