Assignment #5

Problem Solving and Programming in C++
Department of Computer Science
Old Dominion University

Background:
Stepwise refinement is a low level design technique in which the programmer writes pseudo-code for what the program is supposed to do in steps, expanding non-obvious steps at each iteration of the process. Eventually, the stepwise refinement will be an almost step by step guideline to the implementation of the program.

Objective:
The main objective of this assignment is to assess students' ability to apply the stepwise refinement process to develop a new algorithm and carry that through to the implementation of the program. Implementation must follow the top-down design approach, where the solution starts by describing the general functionality of a game. Next, more details are provided in successive steps to refine the implementation.

Problem Description:
Assume you are hired by a game developing company to write a new computer game for kids. This company has decided to create a version of checkers with a few different rules hoping that this new game will be more entertaining. If you are unfamiliar with the original game of checkers, please learn how to play. Here is a link that includes the instructions: https://www.itsyourturn.com/t_helptopic2030.html

This modified version of checkers that you required to create is very much like the original game. The differences between the game are as follows:

- This new version of checkers is played on a 6x6 board instead of 8x8
- Pieces will be numbered "1" or "2," only pieces with the same number can jump each other
- A player cannot jump more than one piece on a given turn
- Players earn points for jumping, first player to 3 wins
- All pieces can move backwards, therefore, "kings" do not exist
- You are not forced to jump a piece given the opportunity

There are several steps that you will need to implement:

First ask the red player to select a piece to move. Next, make sure that they have selected a valid piece; then request the position they would like to move the piece. Again, make sure that this new position is valid (Ex1. They are not landing on another piece. Ex2. If they are choosing

to jump, the piece being jumped must have the same number as the piece jumping). In addition, give the player the option to pass their turn in case their piece has no valid places to move. After the Red Player's turn is over, validate that they do not have the amount of points to win and print the game board. Now, implement the same process for the black player, alternating turns until a winner has been found. Here is a demonstration of your final game:
https://www.youtube.com/watch?v=Jmxl5e_UdqM

You need to write the stepwise refinement prior to implementing your game so you can show the project manager what you plan to do. You can use the lecture notes to read more about stepwise refinement.

General Instructions:

- You are given a code that compiles with no errors.
- Functions to update the board are already provided, you will need to update the data structures that correspond to places on the board.
- You only need to implement the `playGame()` function using the step-wise refinement approach.
- Your job is to ask the red and black player for their input, then check that they have made a valid decision and update the board using the given data structures.

Task:

Your task is to apply the technique of stepwise refinement to design an algorithm for the *playGame()*function and play the game of modified checkers. Your *playGame()* function should not break the provided working code, and it should interact with and call other given functions.

Help: This assignment requires development of the program using step-wise refinements to write the C++ program which implements the checkers game as described. The game will be played by two human players (*no computer player*). Step-wise refinement is a technique used for writing programs. The process starts with a simple statement describing the main functionality of the program. Thereafter, the programmer repeatedly and gradually expands this statement into two or more statements. Next, the new statement(s) is/are repeatedly expanded into more detailed statement(s) with the goal of moving towards the final implementation of your program.

Use a text editor (*e.g.*, the **Code::Blocks** editor or Notepad) to write the first pseudo-code statement of the problem. Next, add more statements that support the implementation of that first pseudo-code statement. Each time you refine a step, copy and paste the current version of the program design to the end of the text file, then make the change within that pasted version. Stick to the instructor's convention of using a different number of ***s** to indicate the level of expansion applicable to each statement (see example below). When the design is complete, the text file will

contain a complete record of the design process used to reach the final design. The text file you submit may be quite lengthy and should show the entire process. Do not remove anything and do not just submit the final iteration. Below is a partial example of stepwise refinement (Note it is only a partial example and has nothing to do with the program for this assignment.):

```
//Determine whether data set is skewed or not

//Determine whether data set is skewed or not
//* read data set
//* process data
//* determine if data is skewed

//Determine whether data set is skewed or not
//* read data set
 //**while there is still data {
 //**  read data from either file or keyboard
 //**  increment counter to find # data items
 //**}
//* process data
//* determine if data is skewed

//Determine whether data set is skewed or not
//* read data set
 //** while there is still data {
 //**  read data from either file or keyboard
 //**  increment counter to find # data items
 //**}
//* process data
 //** put data in order
 //** determine average
 //** determine standard deviation
 //** determine median
 //** determine skewness
//* determine if data is skewed
```

Initial Problem Statement

Iterations 2, 3, and 4

….(this process would be continued until every statement is adequately refined using pseudo-code)

```
Red Player, select the piece you would like to move(x y): 0 1

Red Player, select the place to move your piece(i.e. x y)(-1 -1 to pass): 1 2


Red Points: 0
Black Points: 0

   ___ ___ ___ ___ ___ ___
  |   |   |   |   |   |   |
5 | b2 |   | b2 |   | b2 |   |
  |___|___|___|___|___|___|
  |   |   |   |   |   |   |
4 |   | b1 |   | b1 |   | b1 |
  |___|___|___|___|___|___|
  |   |   |   |   |   |   |
3 |   |   |   |   |   |   |
  |___|___|___|___|___|___|
  |   |   |   |   |   |   |
2 |   | r1 |   |   |   |   |
  |___|___|___|___|___|___|
  |   |   |   |   |   |   |
1 |   |   | r1 |   | r1 |   |
  |___|___|___|___|___|___|
  |   |   |   |   |   |   |
0 |   | r2 |   | r2 |   | r2 |
  |___|___|___|___|___|___|
    0   1   2   3   4   5

Black Player, select the piece you would like to move(x y):
```

Submission notes:

- Using global variables will result in -10 points off of your final mark.
- Submit the entire project folder which includes all files from your project, especially the .cpp, .h, and .cbp file(s).
- Zip the project folder and name it as "Assg5_cslogin", where the cslogin is your login ID for the computers at the Department of Computer Science at ODU. (To zip the folder, Right click on it and select "send to" and then click "compressed (zipped) folder")
- Submit the zipped file to the respective Blackboard link.