Assignment #5

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

Objectives: 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.

Description:

The traditional game: Many of us are familiar with the traditional "2D Tic-Tac-Toe", which is a paper-and-pencil game for two players "o" and "x". The two players attempt to fill in spaces on a 3x3 grid. The player who places 3 consecutive "o" or "x" in a row, column, or diagonal wins. The following figure displays an example – in which, player "o" won the game.

o	**o**	**o**
x	•	•
x	•	•

The 3D game: The 3D Tic-Tac-Toe is similar to the traditional 2D game, but it is played in a cubical array of cells, *i.e.*, 3x3x3. The game is played by 2 players, and they take turns placing markers in the array. The first player who places three markers in a row, column, or diagonal wins the game. The three layers are:

Top	**Center**	**Bottom**
• • •	• • •	• • •
• • •	• • •	• • •
• • •	• • •	• • •

In this configuration (*3D version*), there are **49** ways to place three markers of the same type in a row, column or diagonal to win the game. The **49** wining cases include the, four distinct diagonals of the cube plus the number of columns, rows and diagonals of the nine distinct layers of the cube without

repetitions. You will not program all possible configurations (**49**) to win the game, however you will only program **33** winning configurations. You can find all winning configurations, which you will program, in the text file **winningPatterns.txt**. For example, one way to win the game, as in the **winningPatterns.txt**, is when a player manages to place a stack of three markers in a row across the different layers (*only win is shown*), as follow:

Top			Center			Bottom		
o	.	.	o	.	.	o	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Another example of a winning case is displayed in the following Figure, where a player placed three markers in the diagonal of the cube.

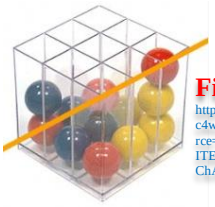

Figure-1

https://www.google.com/search?q=3d+tic+tac+toe&biw=1600&bih=799&tbm=isch&imgil=R0lgPePdjG5iM%253A%253Bc4wcc8rE0Xp1QM%253Bhttp%25253A%25252F%25252Fwww.myhip.com%25252F3d_tic_tac_toe_too_splash.html&source=iu&pf=m&fir=R0lgPenPdjG5iM%253A%252Cc4wcc8rE0Xp1QM%252C_&usg=__CfWmQ9jsAKMsX7TWbKy-ITEhBw94%3D&ved=0ahUKEwjF86ajk7zOAhUFNSYKHZxb-ChAQyjcIMw&ei=leWtV4XVE4XqmAGct6mAAQ#imgrc=__3zEyghYOZFKM%3A

This game might be unfamiliar to you, so ensure to play **3dTicTacToe.exe** (provided) to understand what is expected. After playing the game, and reading the description, you should be able to implement the game.

Task: For this assignment, there will be 2 players: player "o" and player "x". Your task includes:

1. Implementing the 3D board (*cubical array of cells*)
2. Requesting the first player ("o") to place his/her markers "o" in any position, where the input be provided in the form of: **layer row column**

   For example, the input **0 0 0** corresponds to the top layer, first row, and first column respectively. This means the various locations in the board are to be 0-indexed, as follow:

0,0,0	**0,0,1**	**0,0,2**	**1,0,0**	**1,0,1**	**1,0,2**	**2,0,0**	**2,0,1**	**2,0,2**
0,1,0	**0,1,1**	**0,1,2**	**1,1,0**	**1,1,1**	**1,1,2**	**2,1,0**	**2,1,1**	**2,1,2**
0,2,0	**0,2,1**	**0,2,2**	**1,2,0**	**1,2,1**	**1,2,2**	**2,2,0**	**2,2,1**	**2,2,2**

3. Setting the board with the user input after validation. If the input is invalid (*e.g.*, "3, 4, 5"), ask the user to try again until the input is valid.
4. Printing the board
5. Checking if the first player has won, and end the game after announcing win

6. Checking if the board is full, if full restart the game
7. Repeating steps 2 - 6 for the second player "x"

The above outline is a minimum guideline followed by the **3dTicTacToe.exe**

Help: This assignment requires development of the program using step-wise refinements to write the C++ program which implements the 3D Tic-Tac-Toe described. The game will be played by two human player (*no computer player*). Keep the main function as simple as possible. 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)

Submission notes:

- Zip the entire **Code::Blocks** project containing all the .cpp, .h, .cbp files along with text file (named SWrefinement.txt) you created for the pseudo code and name the zipped file "Assg5_cslogin.zip", where the cslogin is your login ID for the computers at the Department of Computer Science at ODU.
- Submit the zipped file using the appropriate Blackboard link.

Important Note: **winningPatterns.txt** is a reference to the program winning patterns.

Sample game: