Name: ______________________________

125 pts

Class Day / Time: ______________________________

Due Date: ______________________________

Assignment #2
Multi-Dimensional Array - Tic Tac Toe

Write a program that will allow **two users** or **one user and the computer** to play the game Tic Tac Toe.

REQUIREMENTS

1. The program should provide the **user with instructions** on how to play and enter data.
   The players should be given the option to exit out after the instructions if they choose not to play.
2. The program should prompt a **main menu** to the players to allow for the following selections:
   a. Exit
   b. Set Player Names
   c. Play in Two Player Mode
   d. Play in One Player Mode
3. **Set Player Names** Option: The program should prompt the players for their names and either assign a token to each player (X or O) or allow them to choose which token they would like to use.
4. **Play in Two Player Mode**
   a. The program should allow the player to make their move by specifying a row and a col.
   For example if Joe wanted to play in row 3 column 3 the I/O would look something like this:
   "Joe's turn! What's your play?: 3 3"
   b. After each turn it should prompt the player by name. For example, when the prompt comes up for a player's turn it should prompt by their name as opposed to "X's turn" – it should say "Joe's turn".
   c. The program should output which player won at the end of the game (by name) or if the players tied. The program will return to the main menu.
5. **Play in One Player Mode**
   a. In this option, the first player will play against the computer, i.e., the second player is your program. The user interface used for the first player is the same described above (used for the two-player mode). The program will return to the main menu once the play ends.
   b. Your program will need to implement an algorithm to determine what position the computer will be playing at each turn. You could for instance, use a random number generator to determine the computer's play. However, this may not be the smarter way for the computer to play. Extra credit may be award for more complex algorithms. You will need to clearly document the algorithm you select for determining the computer's play, to be included in the program documentation.
6. **Modify the display function** as follows:
   - Clear the screen each time it is displayed – to do this you will need to use the the system("cls"); command and #include <cstdlib> (for the mac this command will be system("clear"); and you'll need to #include <cursor.h>) .
   - Fully document this function to the degree that I will know you understand how it works.
   - Modify the single letter variable names to something more descriptive.

You will need to run the .exe file directly (rather than through the Eclispe console for this to work properly). To run the .exe file directly go into the workspace folder for you project. Open the debug folder and double click on the ".exe" file.

Use the given header file with the descriptions of the functions. **Do not add or modify the parameters → except you should add constants for the row and column size to use when in the parameter lists when you pass the board and throughout the program**. You may add additional functions that you feel are necessary to break the code down – but it is not necessary. Be sure to test your code thoroughly (i.e. every possible winning condition as well as tied conditions).

On the last page I have included the DisplayBoard function which you will use in your program – be sure to modify it as stated above (#6).

You program must be demonstrated to receive credit.

Turn in (IN THIS ORDER)

1. The first page of this lab
2. Your header file
3. The listing of main.cpp (conforming to style discussed in class)
4. A listing of your functions in the order provided within the header file (separate from the main.cpp file – don't split functions between pages).
5. The listing of your print heading function – should display above the game instructions.

Header File

```
/* appropriate documentation should go here */

#ifndef TICTACHEADER_H_
#define TICTACHEADER_H_

#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

/****** THE ONLY MODIFICATIONS TO THIS FILE SHOULD BE:                  ******/
/****** 1 - ADDING CONSTANTS FOR THE ARRAY SIZES                         ******/
/****** 2 - UPDATING THE PROTOTYPES TO INCLUDE THE ARRAY SIZE CONSTANT   ******/
/******     DEFINED ABOVE --- NO OTHER MODIFICATIONS TO PARAMETERS!!!    ******/
/****** 3 - REMOVE THESE COMMENTS                                        ******/

/*******************************************************************************
 * OutputInstruct
 *   This function outputs instructions to the users.  There are no input
 *   or output parameters for this function as it only displays text to
 *   the screen.
 *
 * RETURNS: nothing
 * → Displays the instructions to the user
 ******************************************************************************/
void OutputInstruct();


/*******************************************************************************
 * InitBoard
 *   This function initializes each spot in the board to a space ' '.
 *
 * RETURNS: Board initialized with all spaces
 ******************************************************************************/
void InitBoard(char boardAr[][3]);    // OUT - tic tac toe board


/*******************************************************************************
 * DisplayBoard
 *   This function outputs the tic tac toe board including the tokens
 *   played in the proper format (as described below).
 *
 *            1        2        3
 *         [1][1] | [1][2] | [1][3]
 *                |        |
 *    1           |        |
 *                |        |
 *        --------------------------
 *         [2][1] | [2][2] | [2][3]
 *                |        |
 *    2           |        |
 *                |        |
 *        --------------------------
 *         [3][1] | [3][2] | [3][3]
 *                |        |
 *    3           |        |
 *                |        |
 *
 * RETURNS: nothing
 * → outputs the current state of the board
 *************************************************************************/
void DisplayBoard(const char boardAr[][3]); // IN - tic tac toe board


/*************************************************************************
 * GetPlayers
 *   This function prompts the user and gets the input for the players' names.
 *   playerX will always contain the name of the player that is using the X token.
 *   playerO will always contain the name of the player that is using the O token.
 *
 * RETURNS: the players names through the variables playerX and playerO.
 *************************************************************************/
void GetPlayers(string& playerX,   // OUT - player X's name
                string& playerO);  // OUT - player O'x name


// As this was written in class - you need to document this
void GetAndCheckInp(char boardAr[][3], char token, string playerX, string playerO);


/*************************************************************************
 * SwitchToken
 *   This function switches the active player.
 *   It takes in a parameter representing the current player's token
 *   as a character value (either an X or an O) and returns the opposite.
 *   For example, if this function receives an X it returns an O. If it
 *   receives and O it returns and X.
 *
 * RETURNS: the token opposite of the one in which it receives.
 *************************************************************************/
char SwitchToken(char token);  // IN - current player's token ('X' or 'O')


/*************************************************************************
 * CheckWin
 *   This function checks to see if either player has run. Once it is
 *     possible for a win condition to exist, this should run after each a
 *     player makes a play.
 *
 * RETURNS the character value of the player that won or a value that
 *   indicates a tie.
 *************************************************************************/
char CheckWin(const char boardAr[][3]); // IN - tic tac toe board


/*************************************************************************
 * OutputWinner
 *   This function receives as input a character indicating which player won
 *   or if the game was a tie and outputs an appropriate message. This function
 *   does not return anything as it simply outputs the appropriate message to
 *   the screen.
 *
 * RETURNS: nothing
 * → Displays the winner's name
 *************************************************************************/
void OutputWinner(char whoWon,     // IN  - represents the winner or a value
                                   //       indicating a tied game.
                  string playerX,  // OUT - player X's name
                  string playerO); // OUT - player O'x name


#endif /* TICTACHEADER_H_ */
```

```
/*******************************************************************************
 * The following function is provided for you… please desk check it and ensure
 *   that you thoroughly understand it.  MODIFY it as stated below!
 *
 * 1 - Be sure to document the following in detail!
 *      (demonstrate that you understand this code segment).
 * 2 - Modify the variable names to something more appropriate.
 * 3 - Use appropriate constants if necessary.
 *******************************************************************************/

void DisplayBoard(const char boardAr[][3])
{
      int i;
      int j;

      cout << setw(10) << "1" << setw(8) << "2" << setw(9) << "3\n";
      for (i = 0; i < 3; i++)
      {

            cout << setw(7) << "[" << i+1 << "][1] | " << "[" << i+1;
            cout << "][2] | " << "[" << i+1 << "][3]" << endl;
            cout << setw(14) << "|" << setw(9) << "|" << endl;

            for (j = 0; j < 3; j++)
            {
                  switch(j)
                  {
                        case 0:  cout << i + 1 << setw(9) << boardAr[i][j];
                                 cout  << setw(4) << "|";
                                 break;

                        case 1:  cout << setw(4) << boardAr[i][j];
                                 cout << setw(5) << "|";
                                 break;

                        case 2:  cout << setw(4) << boardAr[i][j] << endl;
                                 break;

                        default: cout <<"ERROR!\n\n";
                  }
            }

            cout << setw(14)<< "|" << setw(10) << "|\n";

            if (i != 2)
            {
                  cout << setw(32) << "--------------------------\n";
            }
      }
      cout << endl << endl;
}
```