```
/*******************************************************************************
 * AUTHOR     : Andrew Gharios
 * STUDENT ID : 1449366
 * AS #1      : Multi-Dimensional Array - Tic Tac Toe
 * CLASS      : CS1B
 * SECTION    : M-TH: 5-7:20p
 * DUE DATE   : 6/28/21
 *******************************************************************************/
#ifndef HEADER_H_
#define HEADER_H_
#include <iostream>
#include <iomanip>
#include <string>
#include <stdlib.h>
#include <time.h>
using namespace std;

const int NUM_COLS = 3;
const int NUM_ROWS = 3;
const int AR_SIZE = 9;

/******************************************************************************
 * 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[][NUM_COLS]); // 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
* à outputs the current state of the board
ASSIGNMENT  #2 Multi-DimensionalArrays  – TIC   TAC TOE CS1B
4 of5
***************************************************************************/
void DisplayBoard(const char boardAr[][NUM_COLS]); // 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's name

                        // As this was written in class - you need to document
                     this
void GetAndCheckInp(char boardAr[][NUM_COLS], 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 0. 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[][NUM_COLS], // IN - tic tac toe board
              char token);                    // IN - token of who's playing.

/
  *****************************************************************************
  **
* 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
* à 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

/
  *****************************************************************************
  **
 * PrintHeaderFile
 *   This function will output the header information
 *
******************************************************************************/
void PrintHeaderFile(ostream& output,      // IN  - output datatype.
    string asName,        // IN  - assignment name
    int asNum,             // IN  - assignment number
    string studentName,   // IN  - student's name
    string classInfo,     // IN  - class that is being taken
    char asType,          // IN  - assignment type
    long long studentID); // IN  - student ID

#endif /* HEADER_H_ */

```

```cpp
/*******************************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* AS #2      : Multi-Dimensional Array - Tic Tac Toe
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 6/29/21
*******************************************************************************/
#include "Header.h"

/*******************************************************************************
* Multi-Dimensional Array - Tic Tac Toe
*------------------------------------------------------------------------------
* This program will interact with the user through a menu and allow them to
* play tic tac toe. The user has the option to set players name, play in two
* players and then in single player vs the computer.
*------------------------------------------------------------------------------
* INPUT:
* playerX  : name of player using token X.
* playerO  : name of player using token O.
* menuPick : option pick of menu.
*******************************************************************************/
int main()
{
    /***************************************************************************
    * CONSTANTS
    * --------------------------------------------------------------------------
    * OUTPUT - USED FOR CLASS HEADING
    * --------------------------------------------------------------------------
    * PROGRAMMER : Programmer's Name
    * CLASS    : Student's Course
    * SECTION    : Class Days and Times
    * LAB_NUM    : Lab Number (specific to this lab)
    * LAB_NAME   : Title of the Lab
    ****************************************************************************
```

```
  /

    const string AS_NAME = "Functions and Arrays";
    const int AS_NUM = 1;
    const string STUDENT_NAME = "Andrew Gharios";
    const string CLASS_INFO = "M-Th 5-7:20p";
    const char AS_TYPE = 'A';
    const long long STUDENT_ID = 1449366;

    char   boardAr[NUM_ROWS][NUM_COLS]; // CALC  - Playing board for tictactoe.
    string playerX;                     // IN & OUT - Player using token X.
    string playerO;                     // IN & OUT - Player using token O.
    char   menuPick;                // IN & CALC - Player's menu pick.
    char   token;                       // CALC     - Token being played.
    char   win;                         // CALC & OUT - Match winner.
    int    plays;                       // CALC      - How many rounds user played.
    int    compCol;                     // CALC      - Computer's column value.
    int    compRow;                     // CALC      - Computer's row value.
    bool   winner;                      // CALC      - Condition to check if someone won.

    srand(time(NULL));

    PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO,
        AS_TYPE, STUDENT_ID);

    OutputInstruct();

    cout << "a.   Exit"                      << endl;
    cout << "b.   Set Player Names"          << endl;
    cout << "c.   Play in Two Player Mode"   << endl;
    cout << "d.   Play in One Player Mode"   << endl;

    cout << endl << "Enter option: ";
    cin.get(menuPick);
    cin.ignore(10000, '\n');

    while (menuPick != 'a')
    {
        switch (menuPick)
        {
            case 'b' :
                GetPlayers(playerX, playerO);
                break;

            case 'c' :
                InitBoard(boardAr);
                token  = 'X';
                plays  = 0;
                winner = false;

                while (!winner && plays < 8)
                {
                    DisplayBoard(boardAr);
                    GetAndCheckInp(boardAr, token, playerX, playerO);
                    plays++;

                    if (plays > 4)
                    {
                        win = CheckWin(boardAr, token);

                        if (win == 'X' || win == 'O')
                        {
                            winner = true;
                        }
                    }

                    token = SwitchToken(token);
                }

                if (!winner)
                {
                    win = 'T';
                }

                OutputWinner(win, playerX, playerO);
                break;

            case 'd' :
                InitBoard(boardAr);
                token  = 'X';
                plays  = 0;
                winner = false;

                while (!winner && plays < 8)
                {
                    DisplayBoard(boardAr);
                    if (token == 'X')
                    {
                        GetAndCheckInp(boardAr, token, playerX, playerO);
                    }
                    else
                    {
                        do
                        {
                            compCol = rand() % 3;
                            compRow = rand() % 3;

                            if (!isspace(boardAr[compRow][compCol]))
                            {
                                compCol = rand() % 3;
                                compRow = rand() % 3;
                            }

                        } while (!isspace(boardAr[compRow][compCol]));

                        boardAr[compRow][compCol] = token;

                    }

                    plays++;

                    if (plays > 4)
                    {
                        win = CheckWin(boardAr, token);

                        if (win == 'X' || win == 'O')
                        {
                            winner = true;
                        }
                    }

                    token = SwitchToken(token);
                }

                if (!winner)
                {
                    win = 'T';
                }

                OutputWinner(win, playerX, playerO);
                break;

        }

        cout << "a.   Exit" << endl;
        cout << "b.   Set Player Names" << endl;
        cout << "c.   Play in Two Player Mode" << endl;
        cout << "d.   Play in One Player Mode" << endl;

        cout << endl << "Enter option: ";
        cin.get(menuPick);
        cin.ignore(10000, '\n');

```

```
    }



    return 0;

}
```

```cpp
#include "Header.h"

void OutputInstruct()
{
    cout << "Welcome to Tic Tac Toe!\n" << endl;
    cout << "Select each player's name and token(X or O).";
    cout << "\n Then select if you want to have 2 players or play with the 
      computer.";
    cout << "\nEach round you will be prompted to select a location to input 
      your token.";
    cout << "\nThe winner will be declared at the end. Good luck and enjoy!!!" 
      << endl;
}
```

```cpp
#include "Header.h"

/*******************************************************************************
* InitBoard
* This function initializes each spot in the board to a space ' '.
*
* INPUTS:
*  boardAr - Gameboard.
*
* No outputs.
******************************************************************************/
void InitBoard(char boardAr[][NUM_COLS]) // OUT - tic tac toe board
{
    int row;
    int col;

    for (row = 0; row < NUM_COLS; row++)
    {
        for (col = 0; col < NUM_COLS; col++)
        {
            boardAr[row][col] = ' ';
        }
    }
}
```

```cpp
#include "Header.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;
}
```

```
#include "Header.h"

/*******************************************************************************
* 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.
*
* INPUTS:
*  playerX - Player with token X.
*  playerO - Player with token O.
*
* No outputs.
******************************************************************************/
void GetPlayers(string& playerX, // OUT - player X’s name
                string& playerO) // OUT - player O’x name
{
    cout << "Enter the name of player using X: ";
    getline(cin, playerX);

    cout << "Enter the name of player using O: ";
    getline(cin, playerO);
}
```

```
#include "Header.h"

void GetAndCheckInp(char boardAr[][NUM_COLS],
                    char token,
                    string playerX,
                    string playerO)
{
    int  row;
    int  col;
    bool valid;

    valid = false;

    do
    {
        if (token == 'X')
        {
            cout << playerX;
        }
        else
        {
            cout << playerO;
        }

        cout << "\'s turn! What is your play?: ";
        cin >> row >> col;
        row--;
        col--;
        if (row > NUM_ROWS - 1 || row < 0)
        {
            cout << "Invalid row - Please try again!\n";
        }
        else if (col > NUM_COLS - 1 || col < 0)
        {
            cout << "Invalid column - Please try again!\n";
        }
        else if (!isspace(boardAr[row][col])) // > if( boardAr[row][col] != ' 
          ')
        {
            cout << "That spot is taken already - try again!\n";
        }
        else
        {
            valid = true;
        }
    } while (!valid);

    boardAr[row][col] = token;
    cin.ignore(10000, '\n');

}
```

```cpp
#include "Header.h"

/******************************************************************************
* 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 0. If it
* receives and O it returns and X.
*
* INPUTS:
*  token - Current player's token.
*
* OUTPUTS:
*  token - Opposite player's token.
*****************************************************************************/
char SwitchToken(char token) // IN - current player’s token ('X' or 'O')
{
    if (token == 'X')
    {
        token = 'O';
    }
    else if (token == 'O')
    {
        token = 'X';
    }

    return token;
}
```

```cpp
#include "Header.h"

/******************************************************************************
* 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.
*
* INPUTS:
*  boardAr - Array for the gameboard.
*  token   - player's turn.
*
* OUTPUTS:
*  whoWon  - character representing game winner or tie.
*****************************************************************************/
char CheckWin(const char boardAr[][NUM_COLS], // IN - tic tac toe board
              char token)                     // IN - token of who's playing.
{
    bool rowWin;
    bool colWin;
    bool diaWin;
    char whoWon;

    rowWin = (boardAr[0][0] == boardAr[0][1] && boardAr[0][1] == boardAr[0][2] && boardAr[0][1] != ' ') ||
             (boardAr[1][0] == boardAr[1][1] && boardAr[1][1] == boardAr[1][2] && boardAr[1][1] != ' ') ||
             (boardAr[2][0] == boardAr[2][1] && boardAr[2][1] == boardAr[2][2] && boardAr[2][1] != ' ');

    colWin = (boardAr[0][0] == boardAr[1][0] && boardAr[1][0] == boardAr[2][0] && boardAr[1][0] != ' ') ||
             (boardAr[0][1] == boardAr[1][1] && boardAr[1][1] == boardAr[2][1] && boardAr[1][1] != ' ') ||
             (boardAr[0][2] == boardAr[1][2] && boardAr[1][2] == boardAr[2][2] && boardAr[1][2] != ' ');

    diaWin = (boardAr[0][0] == boardAr[1][1] && boardAr[1][1] == boardAr[2][2] && boardAr[1][1] != ' ') ||
             (boardAr[2][0] == boardAr[1][1] && boardAr[1][1] == boardAr[0][2] && boardAr[1][1] != ' ');




    if (rowWin || colWin || diaWin)
    {
        whoWon = token;
    }
    else
    {
        whoWon = 'N';
    }

    return whoWon;
}
```

```cpp
#include "Header.h"

void OutputWinner(char whoWon, // IN - represents the winner or value 
  indicating tie.
                  string playerX, // OUT - player X’s name
                  string playerO) // OUT - player O’x name
{
    if (whoWon == 'X')
    {
        cout << endl;
        cout << playerX << " is the winner of the game.";
        cout << endl;
    }
    else if (whoWon == 'O')
    {
        cout << endl;
        cout << playerO << " is the winner of the game.";
        cout << endl;
    }
    else
    {
        cout << endl;
        cout << "No winners! It was a tie.";
        cout << endl;
    }
}
```

```cpp
#include "Header.h"

/*************************************************************************
 * PrintHeaderFile
 *   This function will output the header information
 *
 *________________________________________________________________________
 * PRE-CONDITIONS
 *   The following parameters need to have a defined value prior to calling
 *   the function
 *          asName: The name of the assignment given in the course
 *          asNum: The number of the assignment given in the course
 *          studentName: The name of the student writing the code
 *          classInfo: The course name, date, and time of the class
 *          asType: Will either output as a lab or an assignment
 *          studentID: The Identification Number of the student
 *************************************************************************/
void PrintHeaderFile(ostream& output,    // IN - output datatype.
                    string asName,       // IN - assignment name
                    int asNum,           // IN - assignment number
                    string studentName,  // IN - student's name
                    string classInfo,    // IN - class that is being taken
                    char asType,         // IN - assignment type
                    long long studentID) // IN - student ID
{
    output << left;
    output << "***************************************************************
      \n";
    output << "*   PROGRAMMED BY : " << studentName << endl;
    output << "*   " << setw(14) << "STUDENT ID " << ": " << studentID << endl;
    output << "*   " << setw(14) << "CLASS " << ": " << classInfo << endl;
    output << "*   ";

    // PROCESSING - This will adjust setws and format appropriately based
    //              on if this is a lab 'L' or assignment

    if (toupper(asType) == 'L')
    {
        output << "LAB #" << setw(9);
    }
    else
    {
        output << "ASSIGNMENT #" << setw(2);
    }
    output << asNum << ": " << asName << endl;
    output << "***************************************************************";
    output << right << endl;

    return;
}
```