```
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/

#ifndef MYHEADER_H_
#define MYHEADER_H_

#include<iostream>
#include<iomanip>
#include<string>
#include<stdlib.h>
#include<time.h>
#include<curses.h>
#include<cstdlib>
using namespace std;

/*************
 * VARIABLES *
 *************/

const int ROW_SIZE = 3; //PROCESS - error checking the input for row
const int COL_SIZE = 3; //PROCESS - error checking the input for col


/*****************************************************************************
 * PrintHeader
 *      This function outputs the header into the screen.
 *****************************************************************************/
void PrintHeader(const string MY_NAME,
           const string CLASS,
           const string CLASS_TIME,
           const int    ASSIGN_NUM,
           const string ASSIGN_NAME);

/*****************************************************************************
 * 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
                                     // Done

/******************************************************************************
 * 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
                                            // Done

/******************************************************************************
 * GetPlayers
 *  This function prompts the user and gets the input for the players' names.
 *  playerXwill always contain the name of the player that is using the X token.
 *  playerOwill 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
                char   &compToken,
                char   &tokenChoice,
                int    option);
                //Done



/******************************************************************************
 * GetAndCheckInp
 *  This functions gets each player's play and checks if the inputed numbers are
 *  in the domain of the row and column of the game. also it checks if the
 *  row and column in the board is empty or no
```

```
111 *
112 * RETURNS: nothing
113 *          it puts the players token in the boardAr
114 ********************************************************************************/
115 void GetAndCheckInp(char    boardAr[][3],char   token,
116                     string  playerX     ,string playerO,
117                     int option      ,char   tokenChoice,
118                     charcompToken);
119                     //Done
120
121 /*******************************************************************************
122  * SwitchToken
123  *   This function switches the active player.
124  *   It takes in a parameter representing the current player's token
125  *   as a character value (either an X or an O) and returns the opposite.
126  *   For example, if this function receives an X it returns an O. If it
127  *   receives and O it returns and X.
128  *
129  * RETURNS: the token opposite of the one in which it receives.
130  ********************************************************************************/
131 char SwitchToken(char token);  // IN -current player'stoken ('X' or 'O')
132                                // Done
133
134
135 /*******************************************************************************
136  *  CheckWin
137  *   This function checks to see if either player has won. Once it is
138  *   possible for a win condition to exist, this should run after each a
139  *   player makes a play.
140  *
141  * RETURNSthe character value of the player that won or a value that
142  *   indicates a tie.
143  ********************************************************************************/
144 char CheckWin(const char boardAr[][3]);// IN -tic tac toe board
145                                        // Done
146
147
148 /*******************************************************************************
149  * OutputWinner
150  *   This function receives as input a character indicating which player won
151  *   or if the game was a tie and outputs an appropriate message. This function
152  *   does not return anything as it simply outputs the appropriate message to
153  *   the screen.
154  *
155  * RETURNS: nothing
156  * Displays the winner's name
157  ********************************************************************************/
158 void OutputWinner(char &wonPlayer,  // IN  -represents the winner or a value
159                                     //  indicating a tied game.
160                   string playerX,   //OUT -player X's name
161                   string playerO,   //OUT -player O'x name
162                   char   tokenChoice,
163                   char   compToken,
164                   int    option);
165
```

```
166
167
168 #endif /* MYHEADER_H_ */
169
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/

#include "MyHeader.h"
/*****************************************************************************
* Assignment 2
* ---------------------------------------------------------------------------
* This program is a simulation of the tic-tac-toe game. based on the player's
* choice, they can play against the computer, or against a friend.
* If the player tries to play against the computer, randomly, the game starts
* by the player or the computer and the computer has the algorithm to block the
* player and win the game.
*
* If the player decides to start the game with a friend, randomly, the games
* by either of the players by chance. and the players play against each other
* ---------------------------------------------------------------------------
* INPUT  : option      -> play against the computer or a friend
*          tokenChoice -> choice of player to play as 'X' or 'O'
*          playerX     -> Name of the player X
*          playerO     -> Name of the player O
*          row         -> player('s) choice of the row
*          col         -> player('s) choice of the column
*
* ---------------------------------------------------------------------------
* PROCESS: Initializing the board
*          Getting Players name
*          getting players choice of play
*          deciding whose turn is it
*          getting the play from both players
*          checking for the input
*          checking for the win
*
*
* ---------------------------------------------------------------------------
* OUTPUT : Who has won the game
*****************************************************************************/


int main()
{
    /*************************************************************************
    * CONSTANTS
    * ---------------------------------------------------------------------
    * OUTPUT - USED FOR CLASS HEADING
    * ---------------------------------------------------------------------
    * PROGRAMMER  : Programmer's Name
    * CLASS       : Student's Course
    * SECTION     : Class Days and Time
    * LAB_NUM     : Lab Number (specific to this lab)
```

```
56      * LAB_NAME    : Title of the Assignment
57      *****************************************************************************/
58      const string PROGRAMMER = "Ali Eshghi";
59      const string CLASS      = "CS1B";
60      const string SECTION= "MW: 7:30p - 9:50p";
61      const int    ASSIGN_NUM = 2;
62      const string ASSIGN_NAME= "Tic-Tac-Toe game";
63
64      PrintHeader(PROGRAMMER, CLASS, SECTION, ASSIGN_NUM, ASSIGN_NAME);
65
66
67      /*************
68       * VARIABLES *
69       *************/
70
71      string  playerX; //IN & OUT - name of player for token X
72      string  playerO; //IN & OUT - name of player for token O
73      int     option;      //IN & PROCESS - single or multi player
74      int     randToken;   //PROCESS      - random token to start the game
75      chartoken;       //PROCESS      - deciding the turn for the players
76      charwonPlayer;   //PROCESS & OUT- which token won the game
77      char    tokenChoice; //IN & PROCESS - choice of token in single player mode
78      charcompToken;   //PROCESS      - system's token in single player mode
79
80      char boardAr[ROW_SIZE][COL_SIZE]; // PROCESS - 2 dimensional array for game
81
82
83      InitBoard(boardAr);
84      //This function initializes each spot in the board to a space ' '.
85
86
87      OutputInstruct();
88      //This function outputs instructions to the users.
89
90
91      do  // do while loop for continuing the game if the player played the single
92      {   //player mode or the game ended in tie
93
94          /*****************************************************************************
95          * INPUT - gets the input for the option in the menu
96          *****************************************************************************/
97          time(NULL);
98
99          InitBoard(boardAr);
100         //This function initializes each spot in the board to a space ' '.
101
102         cout << "MENU:"             << endl;
103         cout << "---------------"   << endl;
104         cout << "1 - Single Player" << endl;
105         cout << "2 - Multiplayer"   << endl;
106         cout << "0 - Exit the Game" << endl << endl;
107
108         cout << "Enter your option to play: ";
109         cin  >> option;
110         cout << endl << endl;
```

```
111
112         if(option == 1) //option = 1 -> single player mode
113         {
114
115             GetPlayers(playerX, playerO, compToken,tokenChoice, option);
116             //This function prompts the user and gets the input for the players'
117             //names.
118
119             DisplayBoard(boardAr);
120             //This function outputs the tic-tac-toe board including the tokens
121             // played in the proper format
122
123
124             /*************************************************************************
125             * PROCESS - random number generator to get which player starts the
126             *           game
127             *************************************************************************/
128             srand(time(NULL));
129
130             randToken = rand() % 2 + 1;
131
132             if(randToken == 1)
133             {
134                 token = 'X';
135             }
136             else if(randToken == 2)
137             {
138                 token = 'O';
139             }
140
141             wonPlayer = CheckWin(boardAr);
142             //This function checks to see if either player has won.
143
144             while(wonPlayer == 'K') //while loop to keep the game going until
145             {                       //one of the player has won or the game ends
146                                     //in tie
147
148                 GetAndCheckInp(boardAr, token, playerX, playerO,
149                                option, tokenChoice, compToken);
150                 //This functions gets each player's play and checks the inputed
151                 //numbers are in the domain of the row and column of the game.
152
153                 cout << endl;
154
155                 DisplayBoard(boardAr);
156                 //This function outputs the tic-tac-toe board including the
157                 //tokens played in the proper format
158
159                 wonPlayer = CheckWin(boardAr);
160                 //This function checks to see if either player has won.
161
162
163
164                 if(wonPlayer == 'K') //if no one has one the game, or ended in
165                 {                    //tie, the game continues
```

```
166
167                 token = SwitchToken(token);
168                 //This function switches the active player.
169             }
170
171         }
172
173         OutputWinner(wonPlayer, playerX, playerO, tokenChoice, compToken,
174                      option);
175
176     }
177
178
179
180
181
182     else if(option == 2) //option = 1 -> single player mode
183     {
184         GetPlayers(playerX, playerO, compToken ,tokenChoice, option);
185         //This function prompts the user and gets the input for the players'
186         //names.
187
188
189         DisplayBoard(boardAr);
190         //This function outputs the tic-tac-toe board including the tokens
191         // played in the proper format
192
193
194         /*****************************************************************
195         * PROCESS - random number generator to get which player starts the
196         *           game
197         *****************************************************************/
198         srand(time(NULL));
199
200         randToken = rand()% 2 + 1;
201
202         if(randToken == 1)
203         {
204             token = 'X';
205         }
206         else if(randToken == 2)
207         {
208             token = 'O';
209         }
210
211         wonPlayer = CheckWin(boardAr);
212         //This function checks to see if either player has won.
213
214
215         while(wonPlayer == 'K') //while loop to keep the game going until
216                                 //one of the player has won or the game ends
217                                 //in tie
218         {
219
220             GetAndCheckInp(boardAr, token, playerX, playerO,
```

```
221                                 option, tokenChoice, compToken);
222                 //This functions gets each player's play and checks the inputed
223                 //numbers are in the domain of the row and column of the game.
224
225                 cout << endl;
226
227                 DisplayBoard(boardAr);
228                 //This function outputs the tic-tac-toe board including the
229                 //tokens played in the proper format
230
231
232                 wonPlayer = CheckWin(boardAr);
233                 //This function checks to see if either player has won.
234
235
236                 if(wonPlayer == 'K') //if no one has one the game, or ended in
237                 {                    //tie, the game continues
238                     token = SwitchToken(token);
239                 }
240
241             }
242
243
244
245             OutputWinner(wonPlayer, playerX, playerO, tokenChoice, compToken,
246                          option);
247
248
249
250         }
251
252     }while(wonPlayer == 'N' || option != 0 || option == 1);
253
254
255     cout << "Thank you for playing my tic-tac-toe game. Have a Great day";
256
257
258     return 0;
259 }
260
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * 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()
{
    cout << "Welcome to the Tic-Tac-Toe computer game!"     << endl << endl;

    cout << "There are two ways that you could play this game."     << endl;
    cout << "1 - Single player"                                     << endl;
    cout << "2 - Multiplayer"                                 << endl << endl;

    cout << "1 - In the single player mode, you are playing against"    << endl;
    cout << "the computer. and at the end, if the game ended   "        << endl;
    cout << "in tie, you get to play again."            << endl << endl;

    cout << "2 - In the multiplayer mode, you can play this game with"  << endl;
    cout << "a friend and you can compete with each other. you guys"<< endl;
    cout << "put in the names when you choose your tokens, then the"<< endl;
    cout << "Game begins. Good luck to both players"<< endl << endl << endl;

}


```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * 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
{
    /*************
     * VARIABLES *
     *************/

    int i; //PROCESS - for the loop
    int j; //PROCESS - for the loop

    for(i = 0; i < ROW_SIZE; i++)
    {
        for(j = 0; j < COL_SIZE; j++)
        {
            boardAr[i][j] = ' ';
        }
    }

}

```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * 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[][COL_SIZE]) // IN -tic tac toe board
{
    /*************
     * VARIABLES *
     *************/

    int i; //used in loop
    int j; //used in loop

    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)
            {
```

```
56                 case 0: cout << i + 1 << setw(9) << boardAr[i][j];
57                         cout  << setw(4) << "|";
58                         break;
59
60                 case 1: cout << setw(4) << boardAr[i][j];
61                         cout << setw(5) << "|";
62                         break;
63
64                 case 2: cout << setw(4) << boardAr[i][j] << endl;
65                         break;
66
67                 default: cout <<"ERROR!\n\n";
68             }
69         }
70
71         cout << setw(14)<< "|"<< setw(10) << "|\n";
72
73         if(i != 2)
74         {
75             cout << setw(32) << "--------------------------\n";
76         }
77     }
78     cout << endl << endl;
79 }
80
81
82
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * GetPlayers
 *  This function prompts the user and gets the input for the players' names.
 *  playerXwill always contain the name of the player that is using the X token.
 *  playerOwill 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,    //IN & OUT -player X's name
                string &playerO,//IN & OUT -player O'x name
                char   &compToken,
                char   &tokenChoice,
                int     option)
{



    if(option == 1) //single player
    {

        /*************************************************************************
        * INPUT - gets the choice of the token that the player wants to play
        *         and based on that inputs the input name from the user to
        *         the player O (if the player chooses O) or player X (if the
        *         player chooses X)
        *************************************************************************/
        cin.ignore(10000,'\n');

        cout << "Would you like to play as player \'X\' or player \'O\'? ";
        cin.get(tokenChoice);
        cin.ignore(1000,'\n');

        if(toupper(tokenChoice) == 'X')
        {
            cout << "What is the Player's name for token X? ";
            getline(cin,playerX);
            playerO = "system";
            compToken = 'O';
        }
        else if(toupper(tokenChoice) == 'O')
        {
            cout << "What is the Player's name for token O? ";
            getline(cin,playerO);
            playerX = "system";
```

```
56              compToken = 'X';
57          }
58      }
59
60      else if(option == 2) //multi-Player
61      {
62          /*************************************************************************
63          * INPUT - gets the name of both playerX and playerO
64          *************************************************************************/
65          cin.ignore(10000,'\n');
66
67          cout << "What is the Player's name for token X? ";
68          getline(cin,playerX);
69
70          cout << "What is the Player's name for token O? ";
71          getline(cin,playerO);
72
73          cout << endl;
74      }
75
76
77  }
78
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * GetAndCheckInp
 *  This functions gets each player's play and checks if the inputed numbers are
 *  in the domain of the row and column of the game. also it checks if the
 *  row and column in the board is empty or no
 *
 * RETURNS: nothing
 *          it puts the players token in the boardAr
 *****************************************************************************/
void GetAndCheckInp(char   boardAr[][3],
                    char   token,
                    string playerX,
                    string playerO,
                    int    option,
                    char   tokenChoice,
                    char   compToken)
{
    /*************
     * VARIABLES *
     *************/

    bool    valid;          //PROCESS - to exit the loop
    boolxPlayersTurn;   //PROCESS - if player token is X, player's turn
    booloPlayersTurn;   //PROCESS - if player token is O, player's turn

    int     row;        //IN & PROCESS - row input for the token
    int     col;        //IN & PROCESS - col input for the token


    /*****************************************************************************
    * PROCESS - determines whose turn is it and if the input is valid or no
    *****************************************************************************/
    valid = false;

    xPlayersTurn = ((token == toupper(tokenChoice))
                 &&  (toupper(tokenChoice) == 'X'));

    oPlayersTurn = ((token == toupper(tokenChoice))
                 &&  (toupper(tokenChoice) == 'O'));



    if (option == 1) //single player
    {
        do
```

```
56          {
57              /*************************************************************************
58              * INPUT & PROCESS - gets the input for the row and the col from the
59              *                   player and checks if it is valid
60              *************************************************************************/
61              if (xPlayersTurn)
62              {
63                  cout << playerX << "\'s turn! What is your play?: ";
64                  cin  >> row   >> col;
65
66                  row--;
67                  col--;
68
69                  if(row > ROW_SIZE - 1 || row < 0)
70                  {
71                      cout << "Invalid row - Please try again! \n";
72                  }
73                  else if(col > COL_SIZE - 1 || col < 0)
74                  {
75                      cout << "Invalid column - please try again!\n";
76                  }
77                  else if(!isspace(boardAr[row][col]))
78                  {
79                      cout << "That spot is already taken - try again\n";
80                  }
81                  else
82                  {
83                      valid = true;
84
85
86                      boardAr[row][col] = token;
87                  }
88
89
90              }
91
92              else if (oPlayersTurn)
93              {
94                  cout << playerO << "\'s turn! What is your play?: ";
95                  cin  >> row   >> col;
96
97                  row--;
98                  col--;
99
100                 if(row > ROW_SIZE - 1 || row < 0)
101                 {
102                     cout << "Invalid row - Please try again! \n";
103                 }
104                 else if(col > COL_SIZE - 1 || col < 0)
105                 {
106                     cout << "Invalid column - please try again!\n";
107                 }
108                 else if(!isspace(boardAr[row][col]))
109                 {
110                     cout << "That spot is already taken - try again\n";
```

```
111                 }
112                 else
113                 {
114                     valid = true;
115                     boardAr[row][col] = token;
116                 }
117 
118             }
119 
120 
121             /*****************************************************************
122             * PROCESS - if the player plays before computer, the computer puts
123             *           its token based on the position of the last play by the
124             *           player, and if it is the computer's turn to play and
125             *           there is no threat for the computer, the computer tries
126             *           to put the token for winning and if the computer sees
127             *           any threat from the player, it blocks the player.
128             *****************************************************************/
129             else if(!xPlayersTurn && !oPlayersTurn)
130             {
131                 cout << "Master's turn..." << endl;
132 
133 
134 
135                 /*****************************************************************
136                 * PROCESS - there are 24 ways that the computer could win and if
137                 *           the computer sees any of those 24 ways available and
138                 *           to win the game. the computer puts its token to the
139                 *           board.
140                 *****************************************************************/
141 
142 
143                 if((boardAr[0][0] == compToken)
144                 && (boardAr[2][0] == compToken)
145                 &&  isspace(boardAr[1][0])) //1
146                 {
147                     boardAr[1][0] = compToken;
148 
149 
150                 }
151                 else if((boardAr[0][1] == compToken)
152                      && (boardAr[2][1] == compToken)
153                      &&  isspace(boardAr[1][1])) //2
154                 {
155                     boardAr[1][1] = compToken;
156 
157 
158                 }
159                 else if((boardAr[0][2] == compToken)
160                      && (boardAr[2][2] == compToken)
161                      &&  isspace(boardAr[1][2])) //3
162                 {
163                     boardAr[1][2] = compToken;
164 
165 
```

```
166                 }
167                 else if((boardAr[0][0] == compToken)
168                      && (boardAr[0][2] == compToken)
169                      && isspace(boardAr[0][2])) //4
170                 {
171                     boardAr[0][1] = compToken;
172
173
174                 }
175                 else if((boardAr[1][0] == compToken)
176                      && (boardAr[1][2] == compToken)
177                      && isspace(boardAr[1][1])) //5
178                 {
179                     boardAr[1][1] = compToken;
180
181
182                 }
183                 else if((boardAr[2][0] == compToken)
184                      && (boardAr[2][2] == compToken)
185                      && isspace(boardAr[2][1])) //6
186                 {
187                     boardAr[2][1] = compToken;
188
189
190                 }
191                 else if((boardAr[0][0] == compToken)
192                      && (boardAr[2][2] == compToken)
193                      && isspace(boardAr[1][1])) //7
194                 {
195                     boardAr[1][1] = compToken;
196
197
198                 }
199                 else if((boardAr[0][2] == compToken)
200                      && (boardAr[2][0] == compToken)
201                      && isspace(boardAr[1][1])) //8
202                 {
203                     boardAr[1][1] = compToken;
204
205
206                 }
207                 else if((boardAr[1][0] == compToken)
208                      && (boardAr[2][0] == compToken)
209                      && isspace(boardAr[0][0])) //9
210                 {
211                     boardAr[0][0] = compToken;
212
213
214                 }
215                 else if((boardAr[1][1] == compToken)
216                      && (boardAr[2][1] == compToken)
217                      && isspace(boardAr[0][1])) //10
218                 {
219                     boardAr[0][1] = compToken;
220
```

```
221
222                 }
223                 else if((boardAr[1][2] == compToken)
224                      && (boardAr[2][2] == compToken)
225                      && isspace(boardAr[0][2])) //11
226                 {
227                     boardAr[0][2] = compToken;
228
229
230                 }
231                 else if((boardAr[0][1] == compToken)
232                      && (boardAr[0][2] == compToken)
233                      && isspace(boardAr[0][0])) //12
234                 {
235                     boardAr[0][0] = compToken;
236
237
238                 }
239                 else if((boardAr[1][1] == compToken)
240                      && (boardAr[1][2] == compToken)
241                      && isspace(boardAr[1][0])) //13
242                 {
243                     boardAr[1][0] = compToken;
244
245
246                 }
247                 else if((boardAr[2][1] == compToken)
248                      && (boardAr[2][2] == compToken)
249                      && isspace(boardAr[2][0])) //14
250                 {
251                     boardAr[2][0] = compToken;
252
253
254                 }
255                 else if((boardAr[1][1] == compToken)
256                      && (boardAr[2][2] == compToken)
257                      && isspace(boardAr[0][0])) //15
258                 {
259                     boardAr[0][0] = compToken;
260
261
262                 }
263                 else if((boardAr[1][1] == compToken)
264                      && (boardAr[2][0] == compToken)
265                      && isspace(boardAr[0][2])) //16
266                 {
267                     boardAr[0][2] = compToken;
268
269
270                 }
271                 else if((boardAr[0][0] == compToken)
272                      && (boardAr[1][0] == compToken)
273                      && isspace(boardAr[2][0])) //17
274                 {
275                     boardAr[2][0] = compToken;
```

```
276
277
278                 }
279                 else if((boardAr[0][1] == compToken)
280                      && (boardAr[1][1] == compToken)
281                      && isspace(boardAr[2][1])) //18
282                 {
283                     boardAr[2][1] = compToken;
284
285
286                 }
287                 else if((boardAr[0][2] == compToken)
288                      && (boardAr[1][2] == compToken)
289                      && isspace(boardAr[2][2])) //19
290                 {
291                     boardAr[2][2] = compToken;
292
293
294                 }
295                 else if((boardAr[0][0] == compToken)
296                      && (boardAr[0][1] == compToken)
297                      && isspace(boardAr[0][2])) //20
298                 {
299                     boardAr[0][2] = compToken;
300
301
302                 }
303                 else if((boardAr[1][0] == compToken)
304                      && (boardAr[1][1] == compToken)
305                      && isspace(boardAr[1][2])) //21
306                 {
307                     boardAr[1][2] = compToken;
308
309
310                 }
311                 else if((boardAr[2][0] == compToken)
312                      && (boardAr[2][1] == compToken)
313                      &&  isspace(boardAr[2][2])) //22
314                 {
315                     boardAr[2][2] = compToken;
316
317
318                 }
319                 else if((boardAr[0][0] == compToken)
320                      && (boardAr[1][1] == compToken)
321                      &&  isspace(boardAr[2][2])) //23
322                 {
323                     boardAr[2][2] = compToken;
324
325
326                 }
327                 else if((boardAr[0][2] == compToken)
328                      && (boardAr[1][1] == compToken)
329                      &&  isspace(boardAr[2][0])) //24
330                 {
```

```
331                     boardAr[2][0] = compToken;
332
333
334                 }
335                 /*****************************************************************
336                 * PROCESS - there are 24 ways that the computer can feel the
337                 *           threat from the player and if there were no way to
338                 *           win before the player could put the token, the
339                 *           computer puts the token in the place to block the
340                 *           player from the winning
341                 *****************************************************************/
342                 else if((boardAr[0][0] == toupper(tokenChoice))
343                      && (boardAr[2][0] == toupper(tokenChoice))
344                      &&  isspace(boardAr[1][0]))
345                 {
346                     boardAr[1][0] = compToken;
347
348                 }
349                 else if((boardAr[0][1] == toupper(tokenChoice))
350                      && (boardAr[2][1] == toupper(tokenChoice))
351                      && isspace(boardAr[1][1]))
352                 {
353                     boardAr[1][1] = compToken;
354
355                 }
356                 else if((boardAr[0][2] == toupper(tokenChoice))
357                      && (boardAr[2][2] == toupper(tokenChoice))
358                      && isspace(boardAr[1][2]))
359                 {
360                     boardAr[1][2] = compToken;
361
362                 }
363                 else if((boardAr[0][0] == toupper(tokenChoice))
364                      && (boardAr[0][2] == toupper(tokenChoice))
365                      && isspace(boardAr[0][2]))
366                 {
367                     boardAr[0][1] = compToken;
368
369                 }
370                 else if((boardAr[1][0] == toupper(tokenChoice))
371                      && (boardAr[1][2] == toupper(tokenChoice))
372                      && isspace(boardAr[1][1]))
373                 {
374                     boardAr[1][1] = compToken;
375
376                 }
377                 else if((boardAr[2][0] == toupper(tokenChoice))
378                      && (boardAr[2][2] == toupper(tokenChoice))
379                      && isspace(boardAr[2][1]))
380                 {
381                     boardAr[2][1] = compToken;
382
383                 }
384                 else if((boardAr[0][0] == toupper(tokenChoice))
385                      && (boardAr[2][2] == toupper(tokenChoice))
```

```
386                      && isspace(boardAr[1][1]))
387                 {
388                     boardAr[1][1] = compToken;
389 
390                 }
391                 else if((boardAr[0][2] == toupper(tokenChoice))
392                      && (boardAr[2][0] == toupper(tokenChoice))
393                      && isspace(boardAr[1][1]))
394                 {
395                     boardAr[1][1] = compToken;
396 
397                 }
398                 else if((boardAr[1][0] == toupper(tokenChoice))
399                      && (boardAr[2][0] == toupper(tokenChoice))
400                      && isspace(boardAr[0][0]))
401                 {
402                     boardAr[0][0] = compToken;
403 
404                 }
405                 else if((boardAr[1][1] == toupper(tokenChoice))
406                      && (boardAr[2][1] == toupper(tokenChoice))
407                      && isspace(boardAr[0][1]))
408                 {
409                     boardAr[0][1] = compToken;
410 
411                 }
412                 else if((boardAr[1][2] == toupper(tokenChoice))
413                      && (boardAr[2][2] == toupper(tokenChoice))
414                      && isspace(boardAr[0][2]))
415                 {
416                     boardAr[0][2] = compToken;
417 
418                 }
419                 else if((boardAr[0][1] == toupper(tokenChoice))
420                      && (boardAr[0][2] == toupper(tokenChoice))
421                      && isspace(boardAr[0][0]))
422                 {
423                     boardAr[0][0] = compToken;
424 
425                 }
426                 else if((boardAr[1][1] == toupper(tokenChoice))
427                      && (boardAr[1][2] == toupper(tokenChoice))
428                      && isspace(boardAr[1][0]))
429                 {
430                     boardAr[1][0] = compToken;
431 
432                 }
433                 else if((boardAr[2][1] == toupper(tokenChoice))
434                      && (boardAr[2][2] == toupper(tokenChoice))
435                      && isspace(boardAr[2][0]))
436                 {
437                     boardAr[2][0] = compToken;
438 
439                 }
440                 else if((boardAr[1][1] == toupper(tokenChoice))
```

```
441                      && (boardAr[2][2] == toupper(tokenChoice))
442                      && isspace(boardAr[0][0]))
443                 {
444                     boardAr[0][0] = compToken;
445
446                 }
447                 else if((boardAr[1][1] == toupper(tokenChoice))
448                      && (boardAr[2][0] == toupper(tokenChoice))
449                      && isspace(boardAr[0][2]))
450                 {
451                     boardAr[0][2] = compToken;
452
453                 }
454                 else if((boardAr[0][0] == toupper(tokenChoice))
455                      && (boardAr[1][0] == toupper(tokenChoice))
456                      &&  isspace(boardAr[2][0]))
457                 {
458                     boardAr[2][0] = compToken;
459
460                 }
461                 else if((boardAr[0][1] == toupper(tokenChoice))
462                      && (boardAr[1][1] == toupper(tokenChoice))
463                      && isspace(boardAr[2][1]))
464                 {
465                     boardAr[2][1] = compToken;
466
467                 }
468                 else if((boardAr[0][2] == toupper(tokenChoice))
469                      && (boardAr[1][2] == toupper(tokenChoice))
470                      && isspace(boardAr[2][2]))
471                 {
472                     boardAr[2][2] = compToken;
473
474                 }
475                 else if((boardAr[0][0] == toupper(tokenChoice))
476                      && (boardAr[0][1] == toupper(tokenChoice))
477                      && isspace(boardAr[0][2]))
478                 {
479                     boardAr[0][2] = compToken;
480
481                 }
482                 else if((boardAr[1][0] == toupper(tokenChoice))
483                      && (boardAr[1][1] == toupper(tokenChoice))
484                      && isspace(boardAr[1][2]))
485                 {
486                     boardAr[1][2] = compToken;
487
488                 }
489                 else if((boardAr[2][0] == toupper(tokenChoice))
490                      && (boardAr[2][1] == toupper(tokenChoice))
491                      && isspace(boardAr[2][2]))
492                 {
493                     boardAr[2][2] = compToken;
494
495                 }
```

```
496                 else if((boardAr[0][0] == toupper(tokenChoice))
497                      && (boardAr[1][1] == toupper(tokenChoice))
498                      && isspace(boardAr[2][2]))
499                 {
500                     boardAr[2][2] = compToken;
501 
502                 }
503                 else if((boardAr[0][2] == toupper(tokenChoice))
504                      && (boardAr[1][1] == toupper(tokenChoice))
505                      && isspace(boardAr[2][0]))
506                 {
507                     boardAr[2][0] = compToken;
508 
509                 }
510 
511                 /*****************************************************************
512                 * PROCESS - if the computer determines no chance to win or
513                 *           determines no threat from the player, it randomly
514                 *           puts a token on the board.
515                 *****************************************************************/
516                 else
517                 {
518 
519                     srand(time(NULL));
520 
521                     row = rand() % 3 + 1;
522                     col = rand() % 3 + 1;
523 
524 
525                     row--;
526                     col--;
527 
528                     while(!isspace(boardAr[row][col]))
529                     {
530                         srand(time(NULL));
531 
532                         row = rand() % 3 + 1;
533                         col = rand() % 3 + 1;
534 
535 
536                         row--;
537                         col--;
538 
539                     }
540 
541                     if(isspace(boardAr[row][col]))
542                     {
543 
544 
545                         boardAr[row][col] = compToken;
546                         valid = true;
547 
548                     }
549 
550 
```

```
551
552                     else if(!isspace(boardAr[row - 1][col - 1]))
553                     {
554                         while(!isspace(boardAr[row - 1][col - 1]))
555                         {
556                             if(row == 1)
557                             {
558                                 srand(time(NULL));
559
560                                 row = rand() % 2 + 1;
561
562                                 if(row == 1)
563                                 {
564                                     row = 2;
565                                 }
566                                 if(row == 2)
567                                 {
568                                     row = 3;
569                                 }
570
571                                 row--;
572                                 col--;
573
574                                 boardAr[row][col] = compToken;
575                                 valid = true;
576                             }
577
578                             else if(row == 2)
579                             {
580                                 srand(time(NULL));
581
582                                 row = rand() % 2 + 1;
583
584                                 if(row == 1)
585                                 {
586                                     row = 1;
587                                 }
588                                 if(row == 2)
589                                 {
590                                     row = 3;
591                                 }
592
593                                 row--;
594                                 col--;
595
596                                 boardAr[row][col] = compToken;
597                                 valid = true;
598                             }
599
600                             else if(row == 3)
601                             {
602                                 srand(time(NULL));
603
604                                 row = rand() % 2 + 1;
605
```

```
606                         if(row == 1)
607                         {
608                             row = 1;
609                         }
610                         if(row == 2)
611                         {
612                             row = 2;
613                         }
614
615                         row--;
616                         col--;
617
618                         boardAr[row][col] = compToken;
619                         valid = true;
620                     }
621
622
623                     else if(col == 1)
624                     {
625                         srand(time(NULL));
626
627                         row = rand() % 2 + 1;
628
629                         if(col == 1)
630                         {
631                             col = 2;
632                         }
633                         if(row == 2)
634                         {
635                             col = 3;
636                         }
637
638                         row--;
639                         col--;
640
641                         boardAr[row][col] = compToken;
642                         valid = true;
643                     }
644
645                     else if(col == 2)
646                     {
647                         srand(time(NULL));
648
649                         col = rand() % 2 + 1;
650
651                         if(col == 1)
652                         {
653                             col = 1;
654                         }
655                         if(col == 2)
656                         {
657                             col = 3;
658                         }
659
660                         row--;
```

```
661                             col--;
662 
663                             boardAr[row][col] = compToken;
664                             valid = true;
665                         }
666 
667                         else if(col == 3)
668                         {
669                             srand(time(NULL));
670 
671                             col = rand() % 2 + 1;
672 
673                             if(col == 1)
674                             {
675                                 col = 1;
676                             }
677                             if(col == 2)
678                             {
679                                 col = 2;
680                             }
681 
682                             row--;
683                             col--;
684 
685                             boardAr[row][col] = compToken;
686                             valid = true;
687                         }
688                     }
689                 }
690             }
691             valid = true;
692         }
693     }while(!valid);
694 }
695 
696 
697 
698 
699 else if(option == 2) // multiplayer
700 {
701 
702     do
703     {
704         /*************************************************************************
705         * INPUT & PROCESS - gets the input for the row and the col from the
706         *                   players and checks if it is valid
707         *************************************************************************/
708         if(token == 'X')
709         {
710             cout << playerX;
711         }
712         else if(token == 'O')
713         {
714             cout << playerO;
715         }
```

```
716
717             cout << "\'s turn! What is your play?: ";
718             cin  >> row   >> col;
719
720             row--;
721             col--;
722
723             if(row > ROW_SIZE - 1 || row < 0)
724             {
725                 cout << "Invalid row - Please try again! \n";
726             }
727             else if(col > COL_SIZE - 1 || col < 0)
728             {
729                 cout << "Invalid column - please try again!\n";
730             }
731             else if(!isspace(boardAr[row][col]))
732             {
733                 cout << "That spot is already taken - try again\n";
734             }
735             else
736             {
737                 valid = true;
738             }
739         }while(!valid);
740
741         boardAr[row][col] = token;
742         cin.ignore(10000,'\n');
743
744     }
745     //clear();
746 }
747
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.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 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)
{

    if(token == 'X')
    {
        token = 'O';
    }
    else if(token == 'O')
    {
        token = 'X';
    }

    return token;
}

```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/

#include "MyHeader.h"

/*****************************************************************************
 *  CheckWin
 *   This function checks to see if either player has won. Once it is
 *   possible for a win condition to exist, this should run after each a
 *   player makes a play.
 *
 * RETURNSthe character value of the player that won or a value that
 *   indicates a tie.
 *****************************************************************************/

char CheckWin(const char boardAr[][3]) //PROCESS - the function checks boardAr
{

    /*************
     * VARIABLES *
     *************/

    bool    xWinWays; //PROCESS - How token X is considered won
    bool    oWinWays; //PROCESS - How token O is considered won

    int     i; // PROCESS - used in loop
    int     j; // PROCESS - used in loop

    charwonPlayer; // PROCESS - returns the won token

    /*****************************************************************************
    * PROCESS - There are 9 ways that the token X can win the game
    *****************************************************************************/
    xWinWays = (((boardAr[0][0] == 'X')  &&
                 (boardAr[1][0] == 'X')  &&
                 (boardAr[2][0] == 'X')) ||

                ((boardAr[0][1] == 'X')  &&
                 (boardAr[1][1] == 'X')  &&
                 (boardAr[2][1] == 'X')) ||

                ((boardAr[0][2] == 'X')  &&
                 (boardAr[1][2] == 'X')  &&
                 (boardAr[2][2] == 'X')) ||

                ((boardAr[0][0] == 'X')  &&
                 (boardAr[0][1] == 'X')  &&
                 (boardAr[0][2] == 'X')) ||

```

```
56                 ((boardAr[1][0] == 'X')  &&
57                  (boardAr[1][1] == 'X')  &&
58                  (boardAr[1][2] == 'X')) ||
59
60                 ((boardAr[2][0] == 'X')  &&
61                  (boardAr[2][1] == 'X')  &&
62                  (boardAr[2][2] == 'X')) ||
63
64                 ((boardAr[0][0] == 'X')  &&
65                  (boardAr[1][1] == 'X')  &&
66                  (boardAr[2][2] == 'X')) ||
67
68                 ((boardAr[0][2] == 'X')  &&
69                  (boardAr[1][1] == 'X')  &&
70                  (boardAr[2][0] =='X')));
71
72
73     /*************************************************************************
74     * PROCESS - There are 9 ways that the token O can win the game
75     *************************************************************************/
76     oWinWays = (((boardAr[0][0] == 'O')  &&
77                  (boardAr[1][0] == 'O')  &&
78                  (boardAr[2][0] == 'O')) ||
79
80                 ((boardAr[0][1] == 'O')  &&
81                  (boardAr[1][1] == 'O')  &&
82                  (boardAr[2][1] == 'O')) ||
83
84                 ((boardAr[0][2] == 'O')  &&
85                  (boardAr[1][2] == 'O')  &&
86                  (boardAr[2][2] == 'O')) ||
87
88                 ((boardAr[0][0] == 'O')  &&
89                  (boardAr[0][1] == 'O')  &&
90                  (boardAr[0][2] == 'O')) ||
91
92                 ((boardAr[1][0] == 'O')  &&
93                  (boardAr[1][1] == 'O')  &&
94                  (boardAr[1][2] == 'O')) ||
95
96                 ((boardAr[2][0] == 'O')  &&
97                  (boardAr[2][1] == 'O')  &&
98                  (boardAr[2][2] == 'O')) ||
99
100                ((boardAr[0][0] == 'O')  &&
101                 (boardAr[1][1] == 'O')  &&
102                 (boardAr[2][2] == 'O')) ||
103
104                ((boardAr[0][2] == 'O')  &&
105                 (boardAr[1][1] == 'O')  &&
106                 (boardAr[2][0] == 'O')));
107
108
109
110
```

```
111     /*************************************************************************
112     * PROCESS - Initializes the wonPlayer based on which token has won the game
113     *************************************************************************/
114     if(xWinWays)
115     {
116         wonPlayer = 'X';
117     }
118 
119     else if(oWinWays)
120     {
121         wonPlayer = 'O';
122     }
123     else
124     {
125         for(i = 0; i < ROW_SIZE; i++)
126         {
127             for(j = 0; j < COL_SIZE; j++)
128             {
129                 if((boardAr[i][j] == ' ') && (!xWinWays && !oWinWays))
130                 {
131                     wonPlayer = 'K';// For KEEP PLAY
132                 }
133             }
134         }
135     }
136 
137 
138 
139     return wonPlayer;
140 
141 }
142 
143 
144 
145 
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * 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 &wonPlayer,  // 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
                  char   tokenChoice,
                  char   compToken,
                  int    option)
{
    if (option == 1)
    {
        /*************************************************************************
        * OUTPUT - outputs the result of the game(who has one). or the game
        *          ended in tie.
        *************************************************************************/
        if(toupper(tokenChoice) == 'X')
        {
            switch(wonPlayer)
            {
            case 'X': cout << "You have won the Game"
                                              << endl;
            break;
            case 'O': cout << "Master has won the Game;";
                            cout <<" Better luck next time;"
                                 << endl;
            break;
            default:  cout << "The Game ended in tie."       << endl;
                      wonPlayer = 'N';
            }
        }

        else if(toupper(tokenChoice) == 'O')
        {
            switch(wonPlayer)
            {
```

```
56              case 'O': cout << "You have won the Game"
57                                             << endl;
58              break;
59              case 'X': cout << "Master has won the Game;";
60                               cout <<" Better luck next time;"
61                                    << endl;
62              break;
63              default:  cout << "The Game ended in tie."       << endl;
64                        wonPlayer = 'N';
65              }
66          }
67      }
68
69      else if(option == 2)
70      {
71          /*************************************************************************
72          * OUTPUT - outputs the result of the game(who has one). or the game
73          *          ended in tie.
74          *************************************************************************/
75          switch(wonPlayer)
76          {
77          case 'X': cout << playerX << " has won the Game" << endl;
78          break;
79          case 'O': cout << playerO << " has won the Game" << endl;
80          break;
81          default:  cout << "The Game ended in tie."       << endl;
82                    wonPlayer = 'N';
83          }
84      }
85  }
86
```

```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1B
 * SECTION    : MW 7:30pm
 * Assign #2  : tic-tac-toe game (multi-dimensional arrays)
 * DUE DATE   : 19 September 2019
 *****************************************************************************/
#include "MyHeader.h"

/*****************************************************************************
 * PrintHeader
 *      This function outputs the header into the screen.
 *****************************************************************************/

void PrintHeader(const string MY_NAME, //OUT
            const string CLASS,         //OUT
            const string CLASS_TIME,    //OUT
            const int    ASSIGN_NUM,    //OUT
            const string ASSIGN_NAME)   //OUT
{
    cout << left;
    cout << "*************************************************\n"       ;
    cout << "*  PROGRAMMED BY : "    << MY_NAME                        ;
    cout << "\n*  "     << setw(14) << "CLASS"     << ": " << CLASS      ;
    cout << "\n*  "     << setw(14) << "SECTION"   << ": " << CLASS_TIME ;
    cout << "\n*  LAB #"<< setw(9)  << ASSIGN_NUM  << ": " << ASSIGN_NAME;
    cout << "\n*************************************************\n\n"   ;
    cout << right;
}


```