```
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/

#ifndef HEADER_H_
#define HEADER_H_
#include <iostream>
#include <iomanip>
#include <string>
#include <cstdlib>
#include <limits>
#include <ios>
using namespace std;

const int NUM_COLS = 3;

/*****************************************************************************
* PrintHeader
*   This function receives an assignment name, type and number then outputs
*   the appropriate header
*    ==> returns nothing - This will output the class heading.
*****************************************************************************/
 void PrintHeader (ostream &output, // IN/OUT - output file
                   string    asName, //IN - assignment Name - used for output
                   char      asType, //IN - assignment Type
                                     //   - (LAB or ASSIGN) - used for output
                   int       asNum); //IN - assignment Name - used for output


/*****************************************************************************
 * 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
 * à 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).*
 * RETURNS: nothing
 * outputs the current state of the board*/
/*************************************************************************/
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'x name

/*************************************************************************
 * GetAndCheckInp
 * This function determins which spot the user wants to play in. They must type
 * in the row and the column. The function obtains the input and verifies that
 * the row and column are in the range and spot is not taken. assume all the
 * elements in the array were intialized to blank spaces.
 *
 *
 *************************************************************************/
void GetAndCheckInp(char boardAr[][NUM_COLS], // IN - tic tac toe board
                    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 000a 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

/*************************************************************************
 * OutputWinner
```

```
115 * This function receives as input a character indicating which player won
116 * or if the game was a tie and outputs an appropriate message. This function
117 * does not return anything as it simply outputs the appropriate message to
118 * the screen.
119 *
120 * RETURNS: nothing Displays the winner's name
121 ********************************************************************************/
122 void OutputWinner(char whoWon,     // IN - represents the winner or a value
123                                    // indicating a tied game.
124                   string playerX,  // OUT - player X's name
125                   string playerO); // OUT - player O'x name
126
127 /*******************************************************************************
128  *  ComputerPlay
129  *      This function has the computer to chose the best spot to play on the
130  * grid based on the conditions of the grid. It will prevent the user from
131  * making a winning move and it will try to chose the winning move for itself
132  *
133  *      Returns nothing but it will return the column and the row values of the
134  *  chosen move through passing by refrence
135  ********************************************************************************/
136 void ComputerPlay(char boardAr[][NUM_COLS], // IN  - Tic tac toe board
137                   int &row,                 // OUT - Row value of game play
138                   int &col);                // OUT - Col value of game play
139
140 /*******************************************************************************
141  *  MenuOuput
142  *      This function ouputs a menu for the user and allows them to choose
143  *  an option.
144  *
145  *      Returns the character that the user has chosen
146  ********************************************************************************/
147 char MenuOuput();
148
149
150 #endif /* HEADER_H_ */
151
```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 *  Tic Tac Toe
 *  ---------------------------------------------------------------------------
 *      This profram will allow the user to play the game of tic tac toe
 *  with either another player or VS a computer. It will provide the user with
 *  instructions of how to play the game.
 *  The user will enter their name and the token they would like to play with.
 *  ---------------------------------------------------------------------------
 *  INPUT:
 *              Name    : The name of the user
 *              Token   : The token which the user choses to play with (X or O)
 *
 *  OUTPUT:
 *              It outputs a string stating which player has won or if the game
 *              is a tie.
 *****************************************************************************/
int main()
{
/*****************************************************************************
 * CONSTANTS
 *---------------------------------------------------------------------------
 *  USED FOR PROCESSING
 *---------------------------------------------------------------------------
 *      ROW_NUM : The size of the row in the 2d array
 *      COL_NUM : The size of the column in the 2d array
 *****************************************************************************/
    const int ROW_NUM = 3;
    const int COL_NUM = 3;


    char boardAr[ROW_NUM][COL_NUM]; // CALC      - The 2d array for the
                                    //             tic tac toe
    char token;                     // IN & CALC - The token which the user choses
    char winner;                    // CALC      - if there is a winner or not
    bool hasWinner;                 // CALC      - If the game has a winner
                                    //             or a tie
    bool gameOver ;                 // CALC      - if the game is over or
                                    //                there is space left
    string playerX;                 // IN & OUT  - The name of the player
    string playerO;                 // IN & OUT  - The name of the player
    char playerChoice;              // IN & CALC - User inputs of the menu
                                    //             and game they want to play
    bool startPlay;                 // CLAC      - decideds to play game or no

    /**INTIALIZATION**/
    winner = ' ';


```

```
58      /*****************************************************************
59       * OUTPUT - outputs the instruction for the player
60       *****************************************************************/
61      PrintHeader(cout, "Tic Tac Toe", 'A', 2);
62      OutputInstruct();
63
64      do
65      {
66          startPlay= false;
67          playerChoice = MenuOuput();
68      /*****************************************************************
69       * PROCESSING - Gets the name of the user
70       *****************************************************************/
71          if(playerChoice == 'B')
72          {
73              GetPlayers(playerX, playerO);
74          }
75      /*****************************************************************
76       * PROCESSING - One playe mode
77       *****************************************************************/
78          else if(playerChoice == 'D')
79          {
80
81              cout << "\n\nYou chose to play against the computer, you are now "
82                      "player X and the computer is player O!\n"
83                      "Good Luck!\n\n\n";
84
85              if(playerX != "")
86              {
87                  startPlay = true;
88                  playerO = "computer";
89              }
90          }
91      /*****************************************************************
92       * PROCESSING - Two player mood
93       *****************************************************************/
94          else if(playerChoice == 'C')
95          {
96              if(playerX != "" && playerO != "")
97              {
98                  startPlay = true;
99              }
100         }
101
102         if(startPlay)
103         {
104
105             InitBoard(boardAr);
106             DisplayBoard(boardAr);
107
108             token = 'X';
109
110             /**INTIALIZATION**/
111             hasWinner = false;
112             gameOver  = false;
113
114             while(!hasWinner && !gameOver)
```

```
115             {
116 
117                 GetAndCheckInp(boardAr, token, playerX, playerO);
118                 DisplayBoard(boardAr);
119 
120                 winner = CheckWin(boardAr);
121 
122                 if (winner == ' ')
123                 {
124                     token = SwitchToken(token);
125                 }
126                 else
127                 {
128                     if( winner == 'T')
129                     {
130                         gameOver = true;
131                         hasWinner  = false;
132 
133                     }
134                     else
135                     {
136                         hasWinner  = true;
137                     }
138 
139                     OutputWinner(winner, playerX, playerO);
140                 }
141 
142             }//END WHILE
143 
144 
145         }//END IF
146 
147     }while(playerChoice != 'A');
148 
149 
150 
151 
152     return 0;
153 }
154 
```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 *      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
 *****************************************************************************/
void OutputInstruct()
{
    const string MENU_OPTION = "Welcome to Tic-Tac-Toe\n\n"
                               "The object of this game is to get three in a"
                               " row.\nPlease choose a token and enter the space"
                               " you would like to place your token in ( row"
                               " first and then the column)\n"
                               "to enter your data please enter the row number"
                               " space and the column number!\n"
                               "Please enter your name before playing the "
                               "game\n\n";


    cout << MENU_OPTION;
}

```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 *  MenuOuput
 *  ---------------------------------------------------------------------------
 *      This function ouputs a menu for the user and allows them to choose
 *  an option.
 *
 *      Returns the character that the user has chosen
 *  ---------------------------------------------------------------------------
 *  PRE-CONDITIONS
 *                  <NONE ASSIGNED>
 *  POST-CONDITON
 *      Returns the character that the user has chosen
 *****************************************************************************/
char MenuOuput()
{
    char playChoice;        // IN   - users choice
    bool validChoice;       // CALC - Validates users choice

    do
    {
    cout <<   "a. Exit\n"
              "b. Set Player Names\n"
              "c. Play in Two Player Mode\n"
              "d. Play in One Player Mode\n"
              "\nplease enter the mode desired: ";

    cin.get(playChoice);
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << endl;

    playChoice = toupper(playChoice);

    validChoice  =  (playChoice == 'A') ||
                    (playChoice == 'B') ||
                    (playChoice == 'C') ||
                    (playChoice == 'D');

    if(!validChoice)
    {
        cout << "\n**********INVALID CHOICE - Please enter a, b, c or d *****\n"
                "\n";
    }

    }while(!validChoice);

    return playChoice;
}
```

58

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 * InitBoard
 * ---------------------------------------------------------------------------
 * This function initializes each spot in the board to a space ' '.
 *
 * RETURNS: Board initialized with all spaces
 * ---------------------------------------------------------------------------
 * PRE-CONDITION:
 *                  boardAr : The tic tac toe bored
 * POST-CONDITION:
 *                  <none assigned>
 *****************************************************************************/
void InitBoard(char boardAr[][NUM_COLS]) // IN - The tic tac toe bored
{
    int row;        // CALC - the row of the 2d array
    int col;        // CALC - the col of the 2d array

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


}

```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 * DisplayBoard
 * ---------------------------------------------------------------------------
 *      This function outputs the tic tac toe board including the tokens
 * played in the proper format (as described below).*
 *      RETURNS: nothing outputs the current state of the board
 * ---------------------------------------------------------------------------
 * Pre-conditions
 *                  boardAr :
 * Post-conditions
 *                  <NONE>
 *****************************************************************************/
void DisplayBoard(const char boardAr[][3])
{
    int row;        // CALC - The loop control variable
    int col;        // CALC - The loop control variable

    system("CLS");

    cout << setw(10) << "1" << setw(8) << "2" << setw(9) << "3\n";

    for (row = 0; row < 3; row++)
    {
        cout << setw(7) << "[" << row+1 << "][1] | " << "[" << row+1;
        cout << "][2] | " << "[" << row+1 << "][3]" << endl;
        cout << setw(14) << "|" << setw(9) << "|" << endl;
        for (col = 0; col < 3; col++)
        {
            switch(col)
            {
                case 0: cout << row + 1 << setw(9) << boardAr[row][col];
                 cout << setw(4) << "|";
                 break;
                case 1: cout << setw(4) << boardAr[row][col];
                 cout << setw(5) << "|";
                 break;
                case 2: cout << setw(4) << boardAr[row][col] << endl;
                 break;
                default: cout <<"ERROR!\n\n";
            }//End switch

        }//END second for loop

        cout << setw(14)<< "|" << setw(10) << "|\n";
        if (row != 2)
        {
            cout << setw(32) << "-------------------------\n";
        }//END if
```

```
58      }//End first for loop
59      cout << endl << endl;
60
61  }
62
```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#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.
 *
 *      RETURNS: the players names through the variables playerX and playerO.
 * ---------------------------------------------------------------------------
 * PRE-CONDITIONS :
 *              the following variables need to be passed by refrence
 *              playerX : player X's name
 *              playerO : player O's name
 *
 *  POST-CONDITION :
 *      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 token;     // IN   - The players character
    string name;    // IN   - The name of the player
    bool valid;     // CALC - Validates the correct input

    cout << "Please enter your name: ";
    getline(cin, name);

    do
    {
        cout << "Which token would you like to use? ( X or O): ";
        cin.get(token);
        cin.ignore(1000, '\n');

        token = toupper(token);

        valid = (token == 'X') || (token == 'O');
        if(!valid)
        {
            cout << "\n********* INVALID INPUT - Please enter X or O ******\n\n";
        }
    }while(!valid);

    if(token == 'X')
    {
        playerX = name;
    }
    else
    {
```

```
58          playerO = name;
59      }
60
61      cout << "Please enter your name: ";
62      getline(cin, name);
63
64      if(token == 'X')
65      {
66          playerO = name;
67      }
68      else
69      {
70          playerX = name;
71      }
72
73      cout << endl << endl;
74
75  }
76
```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 *  GetAndCheckInp
 *  ---------------------------------------------------------------------------
 *      This function determins which spot the user wants to play in. They must
 * type in the row and the column. The function obtains the input and verifies
 * that the row and column are in the range and spot is not taken. assume all
 * the elements in the array were intialized to blank spaces.
 *
 *      RETURNS NOTHING
 *  ---------------------------------------------------------------------------
 *  PRE-CONDITON:
 *                  boardAr
 *                  token
 *                  playerX
 *                  playerO
 *  POST-CONDITION:
 *                  RETURNS NOTHING
 *****************************************************************************/
void GetAndCheckInp( char baordAr[][NUM_COLS],
                     char token,
                     string playerX,
                     string playerO)
{
    int row;        // CALC - Checks the rows of the 2d array
    int col;        // CALC - Checks the columns of the 2d array
    bool valid;     // CALC - Validates the correct spot
    char space;     // CALC - The empty spaces on the board
    bool doIgnore;  // CALC - Clears input buffer



    /**INTIALIZATION**/
    space = ' ';
    valid = false;
    doIgnore = false;

    do
    {
        if(token == 'X')
        {
            cout << playerX;
        }
        else
        {
            cout << playerO;
        }
        cout << "\'s turn! what is your play?: ";

```

```
58          if(playerO == "computer" && token == 'O')
59          {
60              //PROCESSING - gets computer to play
61              ComputerPlay(baordAr, row, col);
62              cout << row+1 << " " << col+1 << endl;
63          }
64          else
65          {
66              doIgnore = true;
67              cin  >> row >> col;
68              row--;
69              col--;
70          }//END IF ELSE IF
71
72          //VALIDATING THE CORRCT INPUT BY THE USER
73          if(row > NUM_COLS-1 || row < 0)
74          {
75              cout << "invalid row - please try again\n";
76          }
77          else if (col > NUM_COLS-1 || col < 0)
78          {
79              cout << "invalid column - please try again\n";
80          }
81          else if(baordAr[row][col] != space)
82          {
83              cout << "This space is taken - please try again\n";
84          }
85          else
86          {
87              valid = true;
88          }//END IF ELSE IF
89
90
91      }while(!valid);
92
93      baordAr[row][col] = token;
94
95      if (doIgnore)
96      {
97          cin.ignore(numeric_limits<streamsize>::max(), '\n');
98      }//END WHILE
99
100 }
101
102
103
```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#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 O. If it
 * receives and O it returns and X.
 *
 *      RETURNS: the token opposite of the one in which it receives.
 * ---------------------------------------------------------------------------
 * PRE-CONDITIONS:
 *                  token :  current player’s token ('X' or 'O')
 * POST-CONDITION:
 *      RETURNS: the token opposite of the one in which it receives.
 *****************************************************************************/
char SwitchToken(char token)    // IN - current player’s token ('X' or 'O')
{
    char newToken;              // CALC - The token that will be switched to

    newToken = (token == 'X'? 'O'
                            : 'X');

    return newToken;
}

```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#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.
 *
 *      RETURNS the character value of the player that won or a value that
 * indicates a tie.
 *  ---------------------------------------------------------------------------
 *  PRE-CONDITON:
 *                  boardAr
 *  POST-CONDITION:
 *              RETURNS the character value of the player that won or a value
 *               that indicates a tie.
 *****************************************************************************/
char CheckWin(const char boardAr[][NUM_COLS])
{
    char whoWon;        // CALC - The char being returned as winner
    bool gameOver;      // CALC - checks to see if any player has won game or
                        //        tie
    bool winner;        // CLAC - checks for winner
    int i;              // CLAC - Loop control variable
    int j;              // CLAC - Loop control variable


    //INTIALIZING
    whoWon   = ' ';
    gameOver = true;
    winner   = false;

    //Checks for wins in every row
    for (i= 0; i < NUM_COLS; i++)
    {
        if ( boardAr[0][i]== boardAr[1][i] &&
             boardAr[0][i]== boardAr[2][i] &&
             boardAr[0][i] != ' ')
        {
            winner = true;
            whoWon = boardAr[0][i];
        }
    }

    //Checks for winner in every column
    if(!winner)
    {
        for (j=0; j < NUM_COLS; j++)
        {
```

```
58              if(boardAr[j][0] == boardAr[j][1] &&
59                 boardAr[j][0] == boardAr[j][2] &&
60                 boardAr[j][0] != ' ')
61              {
62                  winner = true;
63                  whoWon = boardAr[j][0];
64              }
65          }
66      }
67
68      //Checks for winners in a diagnol
69      if(!winner)
70      {
71          if(boardAr[0][0] == boardAr[1][1] &&
72             boardAr[0][0] == boardAr[2][2] &&
73             boardAr[0][0] != ' ')
74          {
75              winner = true;
76              whoWon = boardAr[0][0];
77          }
78      }
79
80      //Checks for second diagnol
81      if(!winner)
82      {
83          if(boardAr[0][2] == boardAr[1][1] &&
84             boardAr[0][2] == boardAr[2][0] &&
85             boardAr[0][2] != ' ')
86          {
87              winner = true;
88              whoWon = boardAr[0][2];
89          }
90      }
91
92      gameOver = false;
93      //Checks if the game has empty spaces
94      if(!winner)
95      {
96          gameOver = true;
97          for (i= 0; i < NUM_COLS; i++)
98          {
99              for (j=0; j < NUM_COLS; j++)
100             {
101                 if (  boardAr[i][j] == ' ')
102                 gameOver = false;
103             }
104         }
105     }
106     if(gameOver)
107     {
108         whoWon = 'T';
109     }
110     return whoWon;
111 }
112
```

```cpp
/*****************************************************************************
 * AUTHOR       : Negin Mashhadi
 * STUDENT ID   : 1084104
 * Assignment#2 : Tic Tac Toe
 * CLASS        : CS1B
 * SECTION      : MW 6:30p
 * DUE DATE     : 02/14/18
 *****************************************************************************/
#include "header.h"
/*****************************************************************************
 * computerPlay
 * ---------------------------------------------------------------------------
 *      This function has the computer to chose the best spot to play on the
 * grid based on the conditions of the grid. It will prevent the user from
 * making a winning move and it will try to chose the winning move for itself
 *
 *  Returns nothing but it will return the column and the row values of the
 *  chosen move through passing by refrence
 *  ---------------------------------------------------------------------------
 *  PRE-CONDITION :
 *                  boardAr : The 2d array of the game
 *              <THE FOLLOWING VARIABLES NEED A PASS BY REFRENCE>
 *                  row     : The row of the 2d array
 *                  col     : The column of the 2d array
 *
 * POST-CONDITION :
 *      Returns nothing but it will return the column and the row values of the
 *  chosen move through passing by refrence
 *****************************************************************************/

void ComputerPlay(char boardAr[][NUM_COLS], // IN - tic tac toe board
                  int &row,
                  int &col)
{
    int randomVal1;     // CALC & OUT - Randomly chosen row move of computer
    int randomVal2;     // CALC & OUT - Randomly chosen row move of computer
    int i;              // CALC       - The loop control variable
    bool validMove;     // CLAC       - Checks if the spot randomly chosen is
                        //              empty
    bool prevent;       // CLAC       - Checks for users winning move to prevent
    bool winGame;       // CALC       - Checks for computers winning move



    winGame   = false;
    prevent   = false;
    validMove = false;


    //PROCESSING - Blocks the user from winning
    for(i=0; i < NUM_COLS; i++)
    {
        if(boardAr[i][0] == boardAr[i][1] && boardAr[i][0] == 'X'
                                          && boardAr[i][2] == ' ')
        {
            prevent = true;
            row = i;
```

```
58              col = 2;
59          }
60          else if(boardAr[i][0] == boardAr[i][2] && boardAr[i][0] == 'X'
61                                                 && boardAr[i][1] == ' ')
62          {
63              prevent = true;
64              row = i;
65              col = 1;
66          }
67          else if(boardAr[i][1] == boardAr[i][2] && boardAr[i][1] == 'X'
68                                                 && boardAr[i][0] == ' ')
69          {
70              prevent = true;
71              row = i;
72              col = 0;
73          }
74          else if(boardAr[0][i] == boardAr[1][i] && boardAr[0][i] == 'X'
75                                                 && boardAr[2][i] == ' ')
76          {
77              prevent = true;
78              row = 2;
79              col = i;
80          }
81          else if(boardAr[1][i] == boardAr[2][i] && boardAr[1][i] == 'X'
82                                                 && boardAr[0][i] == ' ')
83          {
84              prevent = true;
85              row = 0;
86              col = i;
87          }
88          else if(boardAr[0][i] == boardAr[2][i] && boardAr[0][i] == 'X'
89                                                 && boardAr[1][i])
90          {
91              prevent = true;
92              row = 1;
93              col = i;
94          }
95
96          //PROCESSING CHECKS FOR WINNING MOVES
97          else if(boardAr[i][0] == boardAr[i][1] && boardAr[i][0] == 'O'
98                                                 && boardAr[i][2] == ' ')
99          {
100             winGame = true;
101             row = i;
102             col = 2;
103         }
104         else if(boardAr[i][0] == boardAr[i][2] && boardAr[i][0] == 'O'
105                                                && boardAr[i][1] == ' ')
106         {
107             winGame = true;
108             row = i;
109             col = 1;
110         }
111         else if(boardAr[i][1] == boardAr[i][2] && boardAr[i][1] == 'O'
112                                                && boardAr[i][0] == ' ')
113         {
114             winGame = true;
```

```
115             row = i;
116             col = 0;
117         }
118         else if(boardAr[0][i] == boardAr[1][i] && boardAr[0][i] == 'O'
119                                                && boardAr[2][i] == ' ')
120         {
121             winGame = true;
122             row = 2;
123             col = i;
124         }
125         else if(boardAr[1][i] == boardAr[2][i] && boardAr[1][i] == 'O'
126                                                && boardAr[0][i] == ' ')
127         {
128             winGame = true;
129             row = 0;
130             col = i;
131         }
132         else if(boardAr[0][i] == boardAr[2][i] && boardAr[0][i] == 'O'
133                                                && boardAr[1][i])
134         {
135             winGame = true;
136             row = 1;
137             col = i;
138         }
139 
140     }//END FOR
141 
142     //PROCESSING CHECKS TO PREVENT DIAGNOL MOVES
143     if(boardAr[0][0] == boardAr[2][2] && boardAr[0][0] == 'X'
144                                       && boardAr[1][1] == ' ')
145     {
146         prevent = true;
147         row = 1;
148         col = 1;
149     }
150     else if(boardAr[0][0] == boardAr[1][1] && boardAr[0][0] == 'X'
151                                            && boardAr[2][2] == ' ')
152     {
153         prevent = true;
154         row = 2;
155         col = 2;
156     }
157     else if(boardAr[1][1] == boardAr[2][2] && boardAr[1][1] == 'X'
158                                            && boardAr[0][0] == ' ')
159     {
160         prevent = true;
161         row = 0;
162         col = 0;
163 
164     }
165     else if(boardAr[0][2] == boardAr[2][0] && boardAr[0][2] == 'X'
166                                            && boardAr[1][1] == ' ')
167     {
168         prevent = true;
169         row = 1;
170         col = 1;
171     }
```

```
172     else if(boardAr[0][2] == boardAr[1][1] && boardAr[0][2] == 'X'
173                                            && boardAr[2][0] == ' ')
174     {
175         prevent = true;
176         row = 2;
177         col = 0;
178     }
179     else if(boardAr[1][1] == boardAr[2][0] && boardAr[1][1] == 'X'
180                                            && boardAr[0][2] == ' ')
181     {
182         prevent = true;
183         row = 0;
184         col = 2;
185     }
186
187     //PROCESSING CHECKS FOR DIAGONAL WINNING MOVES
188
189     if(boardAr[0][0] == boardAr[2][2] && boardAr[0][0] == 'O'
190                                       && boardAr[1][1] == ' ')
191     {
192         winGame = true;
193         row = 1;
194         col = 1;
195     }
196     else if(boardAr[0][0] == boardAr[1][1] && boardAr[0][0] == 'O'
197                                            && boardAr[2][2] == ' ')
198     {
199         winGame = true;
200         row = 2;
201         col = 2;
202     }
203     else if(boardAr[1][1] == boardAr[2][2] && boardAr[1][1] == 'O'
204                                            && boardAr[0][0] == ' ')
205     {
206         winGame = true;
207         row = 0;
208         col = 0;
209
210     }
211     else if(boardAr[0][2] == boardAr[2][0] && boardAr[0][2] == 'O'
212                                            && boardAr[1][1] == ' ')
213     {
214         winGame = true;
215         row = 1;
216         col = 1;
217     }
218     else if(boardAr[0][2] == boardAr[1][1] && boardAr[0][2] == 'O'
219                                            && boardAr[2][0] == ' ')
220     {
221         winGame = true;
222         row = 2;
223         col = 0;
224     }
225     else if(boardAr[1][1] == boardAr[2][0] && boardAr[1][1] == 'O'
226                                            && boardAr[0][2] == ' ')
227     {
228         winGame = true;
```

```
229         row = 0;
230         col = 2;
231     }
232 
233 
234 
235     if(!prevent)
236     {
237         while(!validMove)
238         {
239             srand(time(NULL));
240             randomVal1 = rand() % 3 ;
241             randomVal2 = rand() % 3 ;
242             if(boardAr[randomVal1][randomVal2] == ' ')
243             {
244                 validMove = true;
245             }
246         }
247 
248         row  = randomVal1;
249         col  = randomVal2;
250     }
251 }
252 
```