```cpp
#pragma once
#include <iostream>
#include <iomanip>
#include <string>
#include <unistd.h>

const unsigned int ONE_SECOND {1000000};

void displayBoard(const char boardAr[][3]);//displays TicTacToe board when called returns nothing
void displayHeader();//displays class header
void displayInstructions();//displays instructions for player
void initBoard(char boardAr[][3]);//initializes TicTacToe board to spaces
void setPlayerNames(std::string &playerX, std::string &playerO);//sets player names defaults will take place if not specified
void getAndCheckInp(const std::string difficulty, const char menuChoice, char boardAr[][3], const char token, const std::string &playerX, const std::string &playerO);
//Get and check input - gets user input and checks to see if spot is take, if move is valid then fills spot with current token
char winLogic(char boardAr[][3]);//checks all 8 win conditions and returns winning token or returns 'c' for continue with game
void switchToken(char &token);//switches token which intern switches player
void displayWinner(const char boardAr[][3], const char &whoWon, const std::string &playerX, const std::string &playerO);//once a player has won this function displays winner and board
bool checkAiInput(const std::string difficulty, char boardAr[][3], const char token);//checks Ai input and depending on what difficulty was selected thats what the Ai's input will be
bool easyAi(char boardAr[][3], const char token);//random move on the board is placed
bool normalAiBlocking(char boardAr[][3], const char token);//specific blocking move is placed if option is available
void thinking(char boardAr[][3]);//displays thinking timer
bool normalWinConditionsAi(char boardAr[][3], const char token);//specific move is win condition is available for Ai
bool hardWinConditionsAi(char boardAr[][3], const char token);//specific move is win condition is available for Ai
bool hardAiBlocking(char boardAr[][3], const char token);//specific blocking move is placed if option is available
```

```cpp
/*****************************************************************************
 * AUTHOR      : Carlos Aguilera
 * STUDENT ID  : 1152562
 * LAB #       : 2
 * CLASS       : CS1B
 * SECTION     : M-W
 * DUE DATE    : 03.04.22
 *****************************************************************************/

#include "main.h"

/*****************************************************************************
 * Title: Multi-Dimensional Array-TicTacToe
 * ---------------------------------------------------------------------------
 * This is a clone of Tic Tac Toe, in this program you will be able to set
 * your players name and choose between going against a Ai or 1v1 against
 * another human. You can change the difficulty of the Ai by entering a
 * string in the menu for that, also on another note if you don't supply
 * a name for your players default name will be put in place.
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * std::string playerX IN & OUT - used to set name for player with token X
 * std::string playerO IN & OUT - used to set name for player with token O
 * char menuChoice IN - used to get choice of menu selection
 * int ignoreCounter CALC - used to ignore cin.ignore once in the program
 * char boardAr[3][3] IN & OUT & CALC - used to store values like token & space
 * char whoWon CALC - used to store token on who won either X or O
 * char token CALC - used to decide winner and used by boardAr for placement move
 * std::string difficultyInput IN & CALC - used for determining Ai difficulty
 * const int ONE_SECOND CALC - used for storing int of 1000000 ms
 *****************************************************************************/
int main()
{
    std::string playerX {"PlayerX"};
    std::string playerO {"PlayerO"};
    char menuChoice {};
    int ignoreCounter {0};
    do
    {
        system("clear");//clear command in the terminal do to wipe every time while loop runs
        displayHeader();
        if(ignoreCounter < 1)
            displayInstructions();
        std::cout << "Main Menu\na. Exit\nb. Set Player Names\nc. Play in Two Player Mode\nd. Play in One Player Mode\nEnter Choice: ";
        if(ignoreCounter > 0 && menuChoice != 'b')//if statement to ignore first run of the while loop
            std::cin.ignore(10, '\n');
        std::cin.get(menuChoice);
        std::cin.ignore(10, '\n');

        switch(menuChoice)
        {
```

```
53              case 'a':
54                  std::cout << "Thank You!\n";
55                  usleep(ONE_SECOND);//Delay from reading next line of code one
    second
56                  system("clear");
57                  break;
58              case 'b':
59                  setPlayerNames(playerX, playerO);
60                  break;
61              case 'c':
62              {
63                  char boardAr[3][3];
64                  initBoard(boardAr);//initialize board to ' '
65                  char whoWon {};
66                  char token {'X'};
67                  std::string difficultyInput {};
68
69                  do
70                  {
71                      system("clear");
72                      displayBoard(boardAr);
73                      getAndCheckInp(difficultyInput, menuChoice, boardAr,
    token, playerX, playerO);
74                      whoWon = winLogic(boardAr);//check win returns token
75                      switchToken(token);
76                  } while (whoWon != 't' && whoWon != 'X' && whoWon != 'O');
77                  displayWinner(boardAr, whoWon, playerX, playerO);//display
    winner
78                  break;
79              }
80              case 'd':
81              {
82                  system("clear");
83                  char boardAr[3][3];
84                  initBoard(boardAr);
85                  char whoWon {};
86                  char token {'X'};
87                  std::string difficultyInput {"Easy"};
88
89                  std::cout << "Select what difficultly you want the AI to
    be!\n";
90                  std::cout << "Easy\nNormal\nHard\nEnter String: ";
91                  std::getline(std::cin, difficultyInput);
92
93                  do
94                  {
95                      system("clear");
96                      displayBoard(boardAr);
97                      getAndCheckInp(difficultyInput, menuChoice, boardAr,
    token, playerX, playerO);
98                      whoWon = winLogic(boardAr);
99                      switchToken(token);
100                 } while (whoWon != 't' && whoWon != 'X' && whoWon != 'O');
101                 displayWinner(boardAr, whoWon, playerX, playerO);
102                 break;
103             }
104         }
105         ++ignoreCounter;
106     }while(menuChoice != 'a');
107     return 0;
```

```
}
```

```cpp
/*******************************************************************************
 * Title: displayBoard
 * ----------------------------------------------------------------------------
 * function displays updated board
 * outer for loop is in charge of displaying this ->
 *    [1][1] | [1][2] | [1][3]
 *           |        |
 * & and controls when this line gets displayed (only twice)
 *    ------------------------
 * & also changing the values
 *
 * inner for loop controls ->
 *  1        |         |
 * ----------------------------------------------------------------------------
 * No Data Table
 * ----------
 *******************************************************************************/

#include "main.h"
using namespace std;

void displayBoard(const char boardAr[][3])
{
    cout << setw(10) << "1" << setw(8) << "2" << setw(9) << "3\n";
    for (int 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 (int j = 0; j < 3; j++)
        {
            switch(j)
            {
                case 0:  cout << i + 1 << setw(9) << boardAr[i][j];
                    cout  << setw(4) << "|";
                    break;

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

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

                default: cout <<"ERROR!\n\n";
            }
        }
        cout << setw(14)<< "|" << setw(10) << "|\n";

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

```

```
/*******************************************************************************
 * Title: displayInstructions
 * ------------------------------------------------------------------------------
 * This function displays the instructions to the player or players
 * returns nothing
 * ------------------------------------------------------------------------------
 * No Data Table
 * ----------
 *******************************************************************************/

#include "main.h"

void displayInstructions()
{
    std::cout << "[Tic Tac Toe]\n";
    std::cout << "<Instructions>\n";
    std::cout << "<p>\nWhen selecting the option to set a name "
              << "playerX is the first to go & is given token 'X'.\n"
              << "If set names option is not selected then default names will
take place.\n</p>\n";
    std::cout << "<p>\n"
<<"                                        How to Play EX. 1
                        \n"
<<"         1       2       3                                      " <<"
1       2       3      \n "
<<"      [1][1] | [1][2] | [1][3]                                    " <<"
[1][1] | [1][2] | [1][3]  \n "
<<"             |       |                                          " <<"
    |        |        \n "
<<"1            |       |                                          " <<"1
     |        |         \n "
<<"             |       |                                          " <<"
    |        |        \n "
<<"     ------------------------                                    " <<"     ---
---------------------- \n "
<<"      [2][1] | [2][2] | [2][3]                                    " <<"
[2][1] | [2][2] | [2][3]  \n "
<<"             |       |                                          " <<"
    |        |        \n "
<<"2            |       |                        ->                " <<"2
     |   X    |         \n "
<<"             |       |                                          " <<"
    |        |        \n "
<<"     ------------------------                                    " <<"     ---
---------------------- \n "
<<"      [3][1] | [3][2] | [3][3]                                    " <<"
[3][1] | [3][2] | [3][3]  \n "
<<"             |       |                                          " <<"
    |        |        \n "
<<"3            |       |                                          " <<"3
     |        |         \n "
<<"             |       |                                          " <<"
    |        |        \n "


<<"\nPlayerX's turn! What's your play? 2 2                         "
<<"PlayerO's turn! What's your play?_ _\n\n\n\n"
```

```
40
41 <<"                                    How not to Play EX. 2
                        \n "
42 <<"                                                        Spot is taken
   on the board! Retry     \n "
43 <<"          1       2       3                            " <<"
   1       2       3      \n "
44 <<"       [1][1] | [1][2] | [1][3]                        " <<"
   [1][1] | [1][2] | [1][3]  \n "
45 <<"              |         |                              " <<"
       |         |         \n "
46 <<"1             |         |                              " <<"1
        |         |         \n "
47 <<"              |         |                              " <<"
       |         |         \n "
48 <<"      ------------------------                         " <<"     ---
   ---------------------- \n "
49 <<"       [2][1] | [2][2] | [2][3]                        " <<"
   [2][1] | [2][2] | [2][3]  \n "
50 <<"              |         |                              " <<"
       |         |         \n "
51 <<"2             |   X     |                    ->        " <<"2
        |   X     |         \n "
52 <<"              |         |                              " <<"
       |         |         \n "
53 <<"      ------------------------                         " <<"     ---
   ---------------------- \n "
54 <<"       [3][1] | [3][2] | [3][3]                        " <<"
   [3][1] | [3][2] | [3][3]  \n "
55 <<"              |         |                              " <<"
       |         |         \n "
56 <<"3             |         |                              " <<"3
        |         |         \n "
57 <<"              |         |                              " <<"
       |         |         \n "
58


59


60 <<"\nPlayerO's turn! What's your play? 2 2                 "
   <<"PlayerO's turn! What's your play?_ _\n</p>\n";
61
62     std::cout << "</Instructions>\n\n";
63 }
```

```c
/*****************************************************************************
 * Title: Initialize Board
 * ---------------------------------------------------------------------------
 * This function initializes board to spaces it has a nested for loop
 * & it assigns every position in the 2D array to a space
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/
#include "main.h"

void initBoard(char boardAr[][3])
{
    for(size_t row {0}; row < 3; row++)
        for(size_t col {0}; col < 3; col++)
            boardAr[row][col] = ' ';
}
```

```cpp
/*****************************************************************************
 * Title: Set Player Names
 * ---------------------------------------------------------------------------
 * Function: takes input from the user and assigns that into playerX and
 * playerO
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * std::string &playerX IN - pass by reference value that takes in player with
 * token X name
 * std::string &playerO IN - pass by reference value that takes in player with
 * token O name
 *****************************************************************************/

#include "main.h"

void setPlayerNames(std::string &playerX, std::string &playerO)
{
    std::cout << "Enter name for player with token X: ";
    std::getline(std::cin, playerX);
    std::cout << "Enter name for player with token O: ";
    std::getline(std::cin, playerO);
}
```

```cpp
/*****************************************************************************
 * Title: Set Player Names
 * ---------------------------------------------------------------------------
 * Function:
 *  Takes input on what position on the board the current player wants to
 *  place current token, if the position on the board is already taken
 *  then it console outputs invalid position and repeats until a position
 *  is valid.
 *  Also calls the checkAiInput function to validate the Ai's input
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * std::string difficulty CALC - to determine what difficulty of Ai
 * const char onePlayerDeterminer CALC - this char is used to determine if
 * the user chose one player mode
 *****************************************************************************/

#include "main.h"

void getAndCheckInp(const std::string difficulty, const char onePlayerDeterminer, char boardAr[][3], const char token, const std::string &playerX, const std::string &playerO)
{
    int row {};
    int col {};
    bool updatedBoard {false};

    do
    {
        if(token == 'X') {
            std::cout << playerX << "'s turn! What's your play? ";
            std::cin >> row >> col;
            --row;
            --col;
        }else if(onePlayerDeterminer != 'd' && token == 'O') {
            std::cout << playerO << "'s turn! What's your play? ";
            std::cin >> row >> col;
            --row;
            --col;
        }else {
            updatedBoard = checkAiInput(difficulty, boardAr, token);//returns true if Ai input was valid
        }

        if(onePlayerDeterminer != 'd' || token == 'X')//if one player mode is active than only on token X will this control flow statement run
        {
            if(isspace(boardAr[row][col])) {//if current element in the 2d array is taken by a space then its available
                boardAr[row][col] = token;
                updatedBoard = true;
            }else {//if its taken by token O or X than its taken
                system("clear");
                std::cout << "Spot is taken on the board! Retry.\n";
                displayBoard(boardAr);
            }
        }
    }while(!updatedBoard);//once board is updated to true this expression will be not true
```

```
54 }
```

```c
/*****************************************************************************
 * Title: Win Logic
 * ---------------------------------------------------------------------------
 * Function:
 *  This function is very simple if you break it down all it does is checks
 *  to see if a win condition is met
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * static int plays CALC - calculates the number of plays made
 *****************************************************************************/

#include "main.h"

char winLogic(char boardAr[][3])
{
    static int plays {0};
    if(plays > 3)
    {
        if(boardAr[0][0] != ' ' && boardAr[0][0] == boardAr[0][1] && boardAr[0][1] == boardAr[0][2]) {
            //same logic applies to if else ladder if position [0][0] does not equal to space, and a win condition of tic tac toe is met then reset plays and return current token
            plays = 0;
            return boardAr[0][0];
        }
        else if(boardAr[1][0] != ' ' && boardAr[1][0] == boardAr[1][1] && boardAr[1][1] == boardAr[1][2]) {
            plays = 0;
            return boardAr[1][0];
        }
        else if(boardAr[2][0] != ' ' && boardAr[2][0] == boardAr[2][1] && boardAr[2][1] == boardAr[2][2]) {
            plays = 0;
            return boardAr[2][0];
        }
        else if(boardAr[0][0] != ' ' && boardAr[0][0] == boardAr[1][0] && boardAr[1][0] == boardAr[2][0]) {
            plays = 0;
            return boardAr[0][0];
        }
        else if(boardAr[0][1] != ' ' && boardAr[0][1] == boardAr[1][1] && boardAr[1][1] == boardAr[2][1]) {
            plays = 0;
            return boardAr[0][1];
        }
        else if(boardAr[0][2] != ' ' && boardAr[0][2] == boardAr[1][2] && boardAr[1][2] == boardAr[2][2]) {
            plays = 0;
            return boardAr[0][2];
        }
        else if(boardAr[0][0] != ' ' && boardAr[0][0] == boardAr[1][1] && boardAr[1][1] == boardAr[2][2]) {
            plays = 0;
            return boardAr[0][0];
        }
        else if(boardAr[0][2] != ' ' && boardAr[0][2] == boardAr[1][1] && boardAr[1][1] == boardAr[2][0]) {
```

```
            plays = 0;
            return boardAr[0][2];
        }
        else if(plays > 7) {
            plays = 0;
            return 't';
        }
    }
    ++plays;
    return 'c';//return c or continue if nothing was met
}
```

```cpp
/*******************************************************************************
 * Title: Switch Token
 * ----------------------------------------------------------------------------
 * Function:
 *  switch token from X to O and O to X
 * ----------------------------------------------------------------------------
 * Data Table
 * ----------
 * char &token CALC - PBR and switch from current token
 ******************************************************************************/

#include "main.h"

void switchToken(char &token)
{
    if(token == 'X')
        token = 'O';
    else if(token == 'O')
        token = 'X';
}
```

```cpp
/*****************************************************************************
 * Title: Display Winner
 * ---------------------------------------------------------------------------
 * Function:
 *  This function takes player that won or takes a condition if it was
 *  a tie game and outputs the results accoordingly
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 *****************************************************************************/

#include "main.h"

void displayWinner(const char boardAr[][3], const char &whoWon, const
std::string &playerX, const std::string &playerO)
{
    if(whoWon == 'X')
    {
        for(size_t i {5}; i > 0; i--)
        {
            system("clear");
            displayBoard(boardAr);
            std::cout << "Congratulations " << playerX << " for winning!!!\n";
            std::cout << "In "<< i << " seconds you will be sent back to the
main menu.\n";
            usleep(ONE_SECOND);
        }
    }else if(whoWon == 'O')
    {
        for(size_t i {5}; i > 0; i--)
        {
            system("clear");
            displayBoard(boardAr);
            std::cout << "Congratulations " << playerO << " for winning!!!\n";
            std::cout << "In "<< i << " seconds you will be sent back to the
main menu.\n";
            usleep(ONE_SECOND);
        }
    }else if(whoWon == 't')
    {
        for(size_t i {5}; i > 0; i--)
        {
            system("clear");
            displayBoard(boardAr);
            std::cout << "TIE Game " << playerX << " & " << playerO << "
Goodluck next time!!!\n";
            std::cout << "In "<< i << " seconds you will be sent back to the
main menu.\n";
            usleep(ONE_SECOND);
        }
    }
}
```

```cpp
/*****************************************************************************
 * Title: Check Ai Input
 * ---------------------------------------------------------------------------
 * Function:
 *  This function takes difficulty and using that logic delegates proper
 *  functions to handle certain difficulty
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 *
 *****************************************************************************/

#include "main.h"
#include <time.h>
#include <stdlib.h>

bool checkAiInput(const std::string difficulty, char boardAr[][3], const char token)
{
    srand(time(NULL));

    if(difficulty == "Easy") {
        return easyAi(boardAr, token);
    }
    else if(difficulty == "Normal") {
        if(normalWinConditionsAi(boardAr, token))
            return true;
        else if(normalAiBlocking(boardAr, token))
            return true;
        else
            return easyAi(boardAr, token);
    }
    else if(difficulty == "Hard") {
        if(hardWinConditionsAi(boardAr, token))
            return true;
        else if(normalWinConditionsAi(boardAr, token))
            return true;
        else if(hardAiBlocking(boardAr, token))
            return true;
        else if(normalAiBlocking(boardAr, token))
            return true;
        else
            return easyAi(boardAr, token);
    }
    return false;
}
```

```cpp
/*****************************************************************************
 * Title: Easy Ai Logic
 * ---------------------------------------------------------------------------
 * Function:
 *  Declares and initializes row and col to a random number between 0 & 2
 *  & then checks to see if spot is available and will continue to loop
 *  until valid
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * int row{rand() % 3} CALC - initializes row to a number between 0 & 2
 * int col{rand() % 3} CALC - initializes col to a number between 0 & 2
 *****************************************************************************/

#include "main.h"

bool easyAi(char boardAr[][3], const char token)
{
    while(true)
    {
        int row {rand() % 3};
        int col {rand() % 3};
        if(isspace(boardAr[row][col])) {
            thinking(boardAr);
            boardAr[row][col] = token;
            return true;
        }
    }
}
```

```
/*****************************************************************************
 * Title: Normal Ai
 * ---------------------------------------------------------------------------
 * Function:
 *  All this function does is reads the board to see if player X has a
 *  possible win condition if so then Ai will block player from that win
 *  condition. Also does a check to see if its already blocked that move
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/

#include "main.h"

bool normalAiBlocking(char boardAr[][3], const char token)
{
    if(boardAr[0][0] == 'X' && boardAr[0][1] == 'X' && boardAr[0][2] != 'O')
{
        //if boardAr at position [0][0] is equal to X and so is [0][1] then
we check if Ai has already blocked [0][2] and if not than Ai blocks
        //same logic applies to the rest of the ladder
        thinking(boardAr);
        boardAr[0][2] = token;
        return true;
    }
    else if(boardAr[0][2] == 'X' && boardAr[0][1] == 'X' && boardAr[0][0] !=
'O') {
        thinking(boardAr);
        boardAr[0][0] = token;
        return true;
    }
    else if(boardAr[1][0] == 'X' && boardAr[1][1] == 'X' && boardAr[1][2] !=
'O') {
        thinking(boardAr);
        boardAr[1][2] = token;
        return true;
    }
    else if(boardAr[1][2] == 'X' && boardAr[1][1] == 'X' && boardAr[1][0] !=
'O') {
        thinking(boardAr);
        boardAr[1][0] = token;
        return true;
    }
    else if(boardAr[2][0] == 'X' && boardAr[2][1] == 'X' && boardAr[2][2] !=
'O') {
        thinking(boardAr);
        boardAr[2][2] = token;
        return true;
    }
    else if(boardAr[2][2] == 'X' && boardAr[2][1] == 'X' && boardAr[2][0] !=
'O') {
        thinking(boardAr);
        boardAr[2][0] = token;
        return true;
    }
    else if(boardAr[0][0] == 'X' && boardAr[1][0] == 'X' && boardAr[2][0] !=
'O') {
        thinking(boardAr);
        boardAr[2][0] = token;
```

```
52          return true;
53      }
54      else if(boardAr[2][0] == 'X' && boardAr[1][0] == 'X' && boardAr[0][0] !=
'O') {
55          thinking(boardAr);
56          boardAr[0][0] = token;
57          return true;
58      }
59      else if(boardAr[0][1] == 'X' && boardAr[1][1] == 'X' && boardAr[2][1] !=
'O') {
60          thinking(boardAr);
61          boardAr[2][1] = token;
62          return true;
63      }
64      else if(boardAr[2][1] == 'X' && boardAr[1][1] == 'X' && boardAr[0][1] !=
'O') {
65          thinking(boardAr);
66          boardAr[0][1] = token;
67          return true;
68      }
69      else if(boardAr[0][2] == 'X' && boardAr[1][2] == 'X' && boardAr[2][2] !=
'O') {
70          thinking(boardAr);
71          boardAr[2][2] = token;
72          return true;
73      }
74      else if(boardAr[2][2] == 'X' && boardAr[1][2] == 'X' && boardAr[0][2] !=
'O') {
75          thinking(boardAr);
76          boardAr[0][2] = token;
77          return true;
78      }
79      else if(boardAr[0][0] == 'X' && boardAr[1][1] == 'X' && boardAr[2][2] !=
'O') {
80          thinking(boardAr);
81          boardAr[2][2] = token;
82          return true;
83      }
84      else if(boardAr[2][2] == 'X' && boardAr[1][1] == 'X' && boardAr[0][0] !=
'O') {
85          thinking(boardAr);
86          boardAr[0][0] = token;
87          return true;
88      }
89      else if(boardAr[0][2] == 'X' && boardAr[1][1] == 'X' && boardAr[2][0] !=
'O') {
90          thinking(boardAr);
91          boardAr[2][0] = token;
92          return true;
93      }
94      else if(boardAr[2][0] == 'X' && boardAr[1][1] == 'X' && boardAr[0][2] !=
'O') {
95          thinking(boardAr);
96          boardAr[0][2] = token;
97          return true;
98      }
99      else
100         return false;
101 }
```

```cpp
/*****************************************************************************
 * Title: Thinking
 * ---------------------------------------------------------------------------
 * Function:
 *  this function outputs thinking
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/

#include "main.h"

void thinking(char boardAr[][3])
{
    for(size_t i {3}; i > 0; i--)
    {
        system("clear");
        displayBoard(boardAr);
        if(i == 3)
            std::cout << "Thinking.\n";
        else if(i == 2)
            std::cout << "Thinking..\n";
        else if(i == 1)
            std::cout << "Thinking...\n";
        usleep(150000);
    }
}
```

```
/*****************************************************************************
 * Title: normalWinConditionsAi
 * ---------------------------------------------------------------------------
 * Function:
 *  this function checks to see if there is a possible win condition for Ai
 *  and if there is then acts on it, normal only does some win possible
 *  win conditions
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/
#include "main.h"

bool normalWinConditionsAi(char boardAr[][3], const char token)
{
    if(boardAr[0][0] == 'O' && boardAr[0][1] == 'O' && boardAr[0][2] != 'X') {
        thinking(boardAr);
        boardAr[0][2] = token;
        return true;
    }
    else if(boardAr[0][2] == 'O' && boardAr[0][1] == 'O' && boardAr[0][0] !=
'X') {
        thinking(boardAr);
        boardAr[0][0] = token;
        return true;
    }
    else if(boardAr[1][0] == 'O' && boardAr[1][1] == 'O' && boardAr[1][2] !=
'X') {
        thinking(boardAr);
        boardAr[1][2] = token;
        return true;
    }
    else if(boardAr[1][2] == 'O' && boardAr[1][1] == 'O' && boardAr[1][0] !=
'X') {
        thinking(boardAr);
        boardAr[1][0] = token;
        return true;
    }
    else if(boardAr[2][0] == 'O' && boardAr[2][1] == 'O' && boardAr[2][2] !=
'X') {
        thinking(boardAr);
        boardAr[2][2] = token;
        return true;
    }
    else if(boardAr[2][2] == 'O' && boardAr[2][1] == 'O' && boardAr[2][0] !=
'X') {
        thinking(boardAr);
        boardAr[2][0] = token;
        return true;
    }
    else if(boardAr[0][0] == 'O' && boardAr[1][0] == 'O' && boardAr[2][0] !=
'X') {
        thinking(boardAr);
        boardAr[2][0] = token;
        return true;
    }
    else if(boardAr[2][0] == 'O' && boardAr[1][0] == 'O' && boardAr[0][0] !=
'X') {
        thinking(boardAr);
```

```
        boardAr[0][0] = token;
        return true;
    }
    else if(boardAr[0][1] == 'O' && boardAr[1][1] == 'O' && boardAr[2][1] !=
'X') {
        thinking(boardAr);
        boardAr[2][1] = token;
        return true;
    }
    else if(boardAr[2][1] == 'O' && boardAr[1][1] == 'O' && boardAr[0][1] !=
'X') {
        thinking(boardAr);
        boardAr[0][1] = token;
        return true;
    }
    else if(boardAr[0][2] == 'O' && boardAr[1][2] == 'O' && boardAr[2][2] !=
'X') {
        thinking(boardAr);
        boardAr[2][2] = token;
        return true;
    }
    else if(boardAr[2][2] == 'O' && boardAr[1][2] == 'O' && boardAr[0][2] !=
'X') {
        thinking(boardAr);
        boardAr[0][2] = token;
        return true;
    }
    else if(boardAr[0][0] == 'O' && boardAr[1][1] == 'O' && boardAr[2][2] !=
'X') {
        thinking(boardAr);
        boardAr[2][2] = token;
        return true;
    }
    else if(boardAr[2][2] == 'O' && boardAr[1][1] == 'O' && boardAr[0][0] !=
'X') {
        thinking(boardAr);
        boardAr[0][0] = token;
        return true;
    }
    else if(boardAr[0][2] == 'O' && boardAr[1][1] == 'O' && boardAr[2][0] !=
'X') {
        thinking(boardAr);
        boardAr[2][0] = token;
        return true;
    }
    else if(boardAr[2][0] == 'O' && boardAr[1][1] == 'O' && boardAr[0][2] !=
'X') {
        thinking(boardAr);
        boardAr[0][2] = token;
        return true;
    }
    else
        return false;
}
```

```c
/*****************************************************************************
 * Title: hardWinConditionsAi
 * ---------------------------------------------------------------------------
 * Function:
 *  This function with the addition of normalWinConditionsAi have every
 *  possible win condition hard coded to be able to make that move and win
 *  the game
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/

#include "main.h"

bool hardWinConditionsAi(char boardAr[][3], const char token)
{
    if(boardAr[0][0] == 'O' && boardAr[0][2] == 'O' && boardAr[0][1] != 'X') {
        thinking(boardAr);
        boardAr[0][1] = token;
        return true;
    }
    if(boardAr[1][0] == 'O' && boardAr[1][2] == 'O' && boardAr[1][1] != 'X') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    if(boardAr[2][0] == 'O' && boardAr[2][2] == 'O' && boardAr[2][1] != 'X') {
        thinking(boardAr);
        boardAr[2][1] = token;
        return true;
    }
    if(boardAr[0][0] == 'O' && boardAr[2][0] == 'O' && boardAr[1][0] != 'X') {
        thinking(boardAr);
        boardAr[1][0] = token;
        return true;
    }
    if(boardAr[0][1] == 'O' && boardAr[2][1] == 'O' && boardAr[1][1] != 'X') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    if(boardAr[0][2] == 'O' && boardAr[2][2] == 'O' && boardAr[1][2] != 'X') {
        thinking(boardAr);
        boardAr[1][2] = token;
        return true;
    }
    if(boardAr[0][0] == 'O' && boardAr[2][2] == 'O' && boardAr[1][1] != 'X') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    if(boardAr[0][2] == 'O' && boardAr[2][0] == 'O' && boardAr[1][1] != 'X') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    else
        return false;
}
```

```
/*****************************************************************************
 * Title: hardAiBlocking
 * ---------------------------------------------------------------------------
 * Function:
 *  Blocks any possible win conditions that player X has
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/

#include "main.h"

bool hardAiBlocking(char boardAr[][3], const char token)
{
    if(boardAr[0][0] == 'X' && boardAr[0][2] == 'X' && boardAr[0][1] != 'O') {
        thinking(boardAr);
        boardAr[0][1] = token;
        return true;
    }
    if(boardAr[1][0] == 'X' && boardAr[1][2] == 'X' && boardAr[1][1] != 'O') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    if(boardAr[2][0] == 'X' && boardAr[2][2] == 'X' && boardAr[2][1] != 'O') {
        thinking(boardAr);
        boardAr[2][1] = token;
        return true;
    }
    if(boardAr[0][0] == 'X' && boardAr[2][0] == 'X' && boardAr[1][0] != 'O') {
        thinking(boardAr);
        boardAr[1][0] = token;
        return true;
    }
    if(boardAr[0][1] == 'X' && boardAr[2][1] == 'X' && boardAr[1][1] != 'O') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    if(boardAr[0][2] == 'X' && boardAr[2][2] == 'X' && boardAr[1][2] != 'O') {
        thinking(boardAr);
        boardAr[1][2] = token;
        return true;
    }
    if(boardAr[0][0] == 'X' && boardAr[2][2] == 'X' && boardAr[1][1] != 'O') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    if(boardAr[0][2] == 'X' && boardAr[2][0] == 'X' && boardAr[1][1] != 'O') {
        thinking(boardAr);
        boardAr[1][1] = token;
        return true;
    }
    else
        return false;
}
```