

```

1 #pragma once
2 #include <iostream>
3 #include <iomanip>
4 #include <string>
5 #include <unistd.h>
6
7 const unsigned int ONE_SECOND {1000000};
8
9 void displayBoard(const char boardAr[][3]); //displays TicTacToe board when
    called returns nothing
10 void displayHeader(); //displays class header
11 void displayInstructions(); //displays instructions for player
12 void initBoard(char boardAr[][3]); //initializes TicTacToe board to spaces
13 void setPlayerNames(std::string &playerX, std::string &playerO); //sets player
    names defaults will take place if not specified
14 void getAndCheckInp(const std::string difficulty, const char menuChoice, char
    boardAr[][3], const char token, const std::string &playerX, const std::string
    &playerO);
15 //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
16 char winLogic(char boardAr[][3]); //checks all 8 win conditions and returns
    winning token or returns 'c' for continue with game
17 void switchToken(char &token); //switches token which intern switches player
18 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
19 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
20 bool easyAi(char boardAr[][3], const char token); //random move on the board is
    placed
21 bool normalAiBlocking(char boardAr[][3], const char token); //specific blocking
    move is placed if option is available
22 void thinking(char boardAr[][3]); //displays thinking timer
23 bool normalWinConditionsAi(char boardAr[][3], const char token); //specific
    move is win condition is available for Ai
24 bool hardWinConditionsAi(char boardAr[][3], const char token); //specific move
    is win condition is available for Ai
25 bool hardAiBlocking(char boardAr[][3], const char token); //specific blocking
    move is placed if option is available

```

```

1  /*****
2  *
3  * AUTHOR      : Carlos Aguilera
4  * STUDENT ID  : 1152562
5  * LAB #       : 2
6  * CLASS       : CS1B
7  * SECTION     : M-W
8  * DUE DATE    : 03.04.22
9  *****/
10 #include "main.h"
11
12 /*****
13 * Title: Multi-Dimensional Array-TicTacToe
14 * -----
15 * This is a clone of Tic Tac Toe, in this program you will be able to set
16 * your players name and choose between going against a Ai or 1v1 against
17 * another human. You can change the difficulty of the Ai by entering a
18 * string in the menu for that, also on another note if you don't supply
19 * a name for your players default name will be put in place.
20 * -----
21 * Data Table
22 * -----
23 * std::string playerX IN & OUT - used to set name for player with token X
24 * std::string playerO IN & OUT - used to set name for player with token O
25 * char menuChoice IN - used to get choice of menu selection
26 * int ignoreCounter CALC - used to ignore cin.ignore once in the program
27 * char boardAr[3][3] IN & OUT & CALC - used to store values like token &
space
28 * char whoWon CALC - used to store token on who won either X or O
29 * char token CALC - used to decide winner and used by boardAr for placement
move
30 * std::string difficultyInput IN & CALC - used for determining Ai difficulty
31 * const int ONE_SECOND CALC - used for storing int of 1000000 ms
32 *****/
33 int main()
34 {
35     std::string playerX {"PlayerX"};
36     std::string playerO {"PlayerO"};
37     char menuChoice {};
38     int ignoreCounter {0};
39     do
40     {
41         system("clear");//clear command in the terminal do to wipe every time
while loop runs
42         displayHeader();
43         if(ignoreCounter < 1)
44             displayInstructions();
45         std::cout << "Main Menu\na. Exit\nb. Set Player Names\nc. Play in Two
Player Mode\nd. Play in One Player Mode\nEnter Choice: ";
46         if(ignoreCounter > 0 && menuChoice != 'b')//if statement to ignore
first run of the while loop
47             std::cin.ignore(10, '\n');
48             std::cin.get(menuChoice);
49             std::cin.ignore(10, '\n');
50
51             switch(menuChoice)
52             {

```

```

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, player0);
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, player0);
74                     whoWon = winLogic(boardAr); //check win returns token
75                     switchToken(token);
76                     } while (whoWon != 't' && whoWon != 'X' && whoWon != '0');
77                     displayWinner(boardAr, whoWon, playerX, player0); //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
91                 std::cout << "Easy\nNormal\nHard\nEnter String: ";
92                 std::getline(std::cin, difficultyInput);
93
94                 do
95                 {
96                     system("clear");
97                     displayBoard(boardAr);
98                     getAndCheckInp(difficultyInput, menuChoice, boardAr,
token, playerX, player0);
99                     whoWon = winLogic(boardAr);
100                     switchToken(token);
101                     } while (whoWon != 't' && whoWon != 'X' && whoWon != '0');
102                     displayWinner(boardAr, whoWon, playerX, player0);
103                     break;
104                 }
105             }
106         ++ignoreCounter;
107     }while(menuChoice != 'a');
108     return 0;

```



```

1  /*****
2  * Title: displayBoard
3  * -----
4  * function displays updated board
5  * outer for loop is in charge of displaying this ->
6  *   [1][1] | [1][2] | [1][3]
7  *       |       |
8  * & and controls when this line gets displayed (only twice)
9  *   -----
10 * & also changing the values
11 *
12 * inner for loop controls ->
13 *   1       |       |
14 * -----
15 * No Data Table
16 * -----
17 *****/
18
19 #include "main.h"
20 using namespace std;
21
22 void displayBoard(const char boardAr[][3])
23 {
24     cout << setw(10) << "1" << setw(8) << "2" << setw(9) << "3\n";
25     for (int i = 0; i < 3; i++)
26     {
27
28         cout << setw(7) << "[" << i+1 << "]" [1] | " << "[" << i+1;
29         cout << "]" [2] | " << "[" << i+1 << "]" [3]" << endl;
30         cout << setw(14) << "|" << setw(9) << "|" << endl;
31
32         for (int j = 0; j < 3; j++)
33         {
34             switch(j)
35             {
36                 case 0: cout << i + 1 << setw(9) << boardAr[i][j];
37                         cout << setw(4) << "|";
38                         break;
39
40                 case 1: cout << setw(4) << boardAr[i][j];
41                         cout << setw(5) << "|";
42                         break;
43
44                 case 2: cout << setw(4) << boardAr[i][j] << endl;
45                         break;
46
47                 default: cout << "ERROR!\n\n";
48             }
49         }
50         cout << setw(14) << "|" << setw(10) << "| \n";
51
52         if (i != 2)
53             cout << setw(32) << "-----\n";
54     }
55     cout << endl << endl;
56 }
57

```

```

1  /*****
2  * Title: displayInstructions
3  * -----
4  * This function displays the instructions to the player or players
5  * returns nothing
6  * -----
7  * No Data Table
8  * -----
9  *****/
10
11 #include "main.h"
12
13 void displayInstructions()
14 {
15     std::cout << "[Tic Tac Toe]\n";
16     std::cout << "<Instructions>\n";
17     std::cout << "<p>\nWhen selecting the option to set a name "
18                 << "playerX is the first to go & is given token 'X'.\n"
19                 << "If set names option is not selected then default names will
take place.\n</p>\n";
20     std::cout << "<p>\n"
21 << "                                How to Play EX. 1
                                \n"
22 << "          1      2      3          " << "
1      2      3      \n "
23 << "          [1][1] | [1][2] | [1][3]          " << "
[1][1] | [1][2] | [1][3] \n "
24 << "          |      |      |          " << "
|      |      | \n "
25 << "1      |      |      |          " << "1
|      |      | \n "
26 << "          |      |      |          " << "
|      |      | \n "
27 << "          -----          " << "          ---
----- \n "
28 << "          [2][1] | [2][2] | [2][3]          " << "
[2][1] | [2][2] | [2][3] \n "
29 << "          |      |      |          " << "
|      |      | \n "
30 << "2      |      |      |          " << "2
|      X      | \n "
31 << "          |      |      |          " << "
|      |      | \n "
32 << "          -----          " << "          ---
----- \n "
33 << "          [3][1] | [3][2] | [3][3]          " << "
[3][1] | [3][2] | [3][3] \n "
34 << "          |      |      |          " << "
|      |      | \n "
35 << "3      |      |      |          " << "3
|      |      | \n "
36 << "          |      |      |          " << "
|      |      | \n "
37
38
39 << "\nPlayerX's turn! What's your play? 2 2          "
<< "Player0's turn! What's your play?_ _\n\n\n\n"

```

```

40
41 <<"                                     \n "
42 <<"                                     \n "
43 <<"                                     \n "
44 <<"                                     \n "
45 <<"                                     \n "
46 <<"                                     \n "
47 <<"                                     \n "
48 <<"                                     \n "
49 <<"                                     \n "
50 <<"                                     \n "
51 <<"                                     \n "
52 <<"                                     \n "
53 <<"                                     \n "
54 <<"                                     \n "
55 <<"                                     \n "
56 <<"                                     \n "
57 <<"                                     \n "
58
59
60 <<"\nPlayer0's turn! What's your play? 2 2
<<"Player0's turn! What's your play?_ _\n</p>\n";
61
62     std::cout << "</Instructions>\n\n";
63 }

```

How not to Play EX. 2

Spot is taken

" <<"

" <<"

" <<"

" <<"1

" <<"

" <<" ---

" <<"

" <<"

" <<"2

" <<"

" <<" ---

" <<"

" <<"

" <<"3

" <<"

"

->

```
1 /*****
2  * Title: Initialize Board
3  * -----
4  * This function initializes board to spaces it has a nested for loop
5  * & it assigns every position in the 2D array to a space
6  * -----
7  * No Data Table
8  * -----
9  *****/
10 #include "main.h"
11
12 void initBoard(char boardAr[][3])
13 {
14     for(size_t row {0}; row < 3; row++)
15         for(size_t col {0}; col < 3; col++)
16             boardAr[row][col] = ' ';
17 }
```



```

1  /*****
2  * Title: Set Player Names
3  * -----
4  * Function: takes input from the user and assigns that into playerX and
5  * player0
6  * -----
7  * Data Table
8  * -----
9  * std::string &playerX IN - pass by reference value that takes in player with
10 * token X name
11 * std::string &player0 IN - pass by reference value that takes in player with
12 * token 0 name
13 *****/
14
15 #include "main.h"
16
17 void setPlayerNames(std::string &playerX, std::string &player0)
18 {
19     std::cout << "Enter name for player with token X: ";
20     std::getline(std::cin, playerX);
21     std::cout << "Enter name for player with token 0: ";
22     std::getline(std::cin, player0);
23 }

```

```

1  /*****
2  * Title: Set Player Names
3  * -----
4  * Function:
5  * Takes input on what position on the board the current player wants to
6  * place current token, if the position on the board is already taken
7  * then it console outputs invalid position and repeats until a position
8  * is valid.
9  * Also calls the checkAiInput function to validate the Ai's input
10 * -----
11 * Data Table
12 * -----
13 * std::string difficulty CALC - to determine what difficulty of Ai
14 * const char onePlayerDeterminer CALC - this char is used to determine if
15 * the user chose one player mode
16 *****/
17
18 #include "main.h"
19
20 void getAndCheckInp(const std::string difficulty, const char
onePlayerDeterminer, char boardAr[][3], const char token, const std::string
&playerX, const std::string &player0)
21 {
22     int row {};
23     int col {};
24     bool updatedBoard {false};
25
26     do
27     {
28         if(token == 'X') {
29             std::cout << playerX << "'s turn! What's your play? ";
30             std::cin >> row >> col;
31             --row;
32             --col;
33         }else if(onePlayerDeterminer != 'd' && token == '0') {
34             std::cout << player0 << "'s turn! What's your play? ";
35             std::cin >> row >> col;
36             --row;
37             --col;
38         }else {
39             updatedBoard = checkAiInput(difficulty, boardAr, token); //returns
true if Ai input was valid
40         }
41
42         if(onePlayerDeterminer != 'd' || token == 'X') //if one player mode is
active than only on token X will this control flow statement run
43         {
44             if(isspace(boardAr[row][col])) { //if current element in the 2d
array is taken by a space then its available
45                 boardAr[row][col] = token;
46                 updatedBoard = true;
47             }else { //if its taken by token 0 or X than its taken
48                 system("clear");
49                 std::cout << "Spot is taken on the board! Retry.\n";
50                 displayBoard(boardAr);
51             }
52         }
53     }while(!updatedBoard); //once board is updated to true this expression will
be not true

```



```

1  /*****
2  * Title: Win Logic
3  * -----
4  * Function:
5  * This function is very simple if you break it down all it does is checks
6  * to see if a win condition is met
7  * -----
8  * Data Table
9  * -----
10 * static int plays CALC - calculates the number of plays made
11 *****/
12
13 #include "main.h"
14
15 char winLogic(char boardAr[][3])
16 {
17     static int plays {0};
18     if(plays > 3)
19     {
20         if(boardAr[0][0] != ' ' && boardAr[0][0] == boardAr[0][1] &&
boardAr[0][1] == boardAr[0][2]) {
21             //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
22             plays = 0;
23             return boardAr[0][0];
24         }
25         else if(boardAr[1][0] != ' ' && boardAr[1][0] == boardAr[1][1] &&
boardAr[1][1] == boardAr[1][2]) {
26             plays = 0;
27             return boardAr[1][0];
28         }
29         else if(boardAr[2][0] != ' ' && boardAr[2][0] == boardAr[2][1] &&
boardAr[2][1] == boardAr[2][2]) {
30             plays = 0;
31             return boardAr[2][0];
32         }
33         else if(boardAr[0][0] != ' ' && boardAr[0][0] == boardAr[1][0] &&
boardAr[1][0] == boardAr[2][0]) {
34             plays = 0;
35             return boardAr[0][0];
36         }
37         else if(boardAr[0][1] != ' ' && boardAr[0][1] == boardAr[1][1] &&
boardAr[1][1] == boardAr[2][1]) {
38             plays = 0;
39             return boardAr[0][1];
40         }
41         else if(boardAr[0][2] != ' ' && boardAr[0][2] == boardAr[1][2] &&
boardAr[1][2] == boardAr[2][2]) {
42             plays = 0;
43             return boardAr[0][2];
44         }
45         else if(boardAr[0][0] != ' ' && boardAr[0][0] == boardAr[1][1] &&
boardAr[1][1] == boardAr[2][2]) {
46             plays = 0;
47             return boardAr[0][0];
48         }
49         else if(boardAr[0][2] != ' ' && boardAr[0][2] == boardAr[1][1] &&
boardAr[1][1] == boardAr[2][0]) {

```

```
50         plays = 0;
51         return boardAr[0][2];
52     }
53     else if(plays > 7) {
54         plays = 0;
55         return 't';
56     }
57 }
58 ++plays;
59 return 'c';//return c or continue if nothing was met
60 }
```

```

1  /*****
2  * Title: Switch Token
3  * -----
4  * Function:
5  *   switch token from X to 0 and 0 to X
6  * -----
7  * Data Table
8  * -----
9  * char &token CALC - PBR and switch from current token
10 *****/
11
12 #include "main.h"
13
14 void switchToken(char &token)
15 {
16     if(token == 'X')
17         token = '0';
18     else if(token == '0')
19         token = 'X';
20 }

```

```

1  /*****
2  * Title: Display Winner
3  * -----
4  * Function:
5  * This function takes player that won or takes a condition if it was
6  * a tie game and outputs the results accordingly
7  * -----
8  * Data Table
9  * -----
10 *****/
11
12 #include "main.h"
13
14 void displayWinner(const char boardAr[][3], const char &whoWon, const
15 std::string &playerX, const std::string &player0)
16 {
17     if(whoWon == 'X')
18     {
19         for(size_t i {5}; i > 0; i--)
20         {
21             system("clear");
22             displayBoard(boardAr);
23             std::cout << "Congratulations " << playerX << " for winning!!!\n";
24             std::cout << "In "<< i << " seconds you will be sent back to the
25             main menu.\n";
26             usleep(ONE_SECOND);
27         }
28     }else if(whoWon == '0')
29     {
30         for(size_t i {5}; i > 0; i--)
31         {
32             system("clear");
33             displayBoard(boardAr);
34             std::cout << "Congratulations " << player0 << " for winning!!!\n";
35             std::cout << "In "<< i << " seconds you will be sent back to the
36             main menu.\n";
37             usleep(ONE_SECOND);
38         }
39     }else if(whoWon == 't')
40     {
41         for(size_t i {5}; i > 0; i--)
42         {
43             system("clear");
44             displayBoard(boardAr);
45             std::cout << "TIE Game " << playerX << " & " << player0 << "
46             Goodluck next time!!!\n";
47             std::cout << "In "<< i << " seconds you will be sent back to the
48             main menu.\n";
49             usleep(ONE_SECOND);
50         }
51     }
52 }

```

```

1  /*****
2  * Title: Check Ai Input
3  * -----
4  * Function:
5  * This function takes difficulty and using that logic delegates proper
6  * functions to handle certain difficulty
7  * -----
8  * Data Table
9  * -----
10 *
11 *****/
12
13 #include "main.h"
14 #include <time.h>
15 #include <stdlib.h>
16
17 bool checkAiInput(const std::string difficulty, char boardAr[][3], const char
token)
18 {
19     srand(time(NULL));
20
21     if(difficulty == "Easy") {
22         return easyAi(boardAr, token);
23     }
24     else if(difficulty == "Normal") {
25         if(normalWinConditionsAi(boardAr, token))
26             return true;
27         else if(normalAiBlocking(boardAr, token))
28             return true;
29         else
30             return easyAi(boardAr, token);
31     }
32     else if(difficulty == "Hard") {
33         if(hardWinConditionsAi(boardAr, token))
34             return true;
35         else if(normalWinConditionsAi(boardAr, token))
36             return true;
37         else if(hardAiBlocking(boardAr, token))
38             return true;
39         else if(normalAiBlocking(boardAr, token))
40             return true;
41         else
42             return easyAi(boardAr, token);
43     }
44     return false;
45 }

```



```

1  /*****
2  * Title: Easy Ai Logic
3  * -----
4  * Function:
5  *   Declares and initializes row and col to a random number between 0 & 2
6  *   & then checks to see if spot is available and will continue to loop
7  *   until valid
8  * -----
9  * Data Table
10 * -----
11 * int row{rand() % 3} CALC - initializes row to a number between 0 & 2
12 * int col{rand() % 3} CALC - initializes col to a number between 0 & 2
13 *****/
14
15 #include "main.h"
16
17 bool easyAi(char boardAr[][3], const char token)
18 {
19     while(true)
20     {
21         int row {rand() % 3};
22         int col {rand() % 3};
23         if(isspace(boardAr[row][col])) {
24             thinking(boardAr);
25             boardAr[row][col] = token;
26             return true;
27         }
28     }
29 }

```

```

1  /*****
2  * Title: Normal Ai
3  * -----
4  * Function:
5  * All this function does is reads the board to see if player X has a
6  * possible win condition if so then Ai will block player from that win
7  * condition. Also does a check to see if its already blocked that move
8  * -----
9  * No Data Table
10 * -----
11 *****/
12
13 #include "main.h"
14
15 bool normalAiBlocking(char boardAr[][3], const char token)
16 {
17     if(boardAr[0][0] == 'X' && boardAr[0][1] == 'X' && boardAr[0][2] != '0')
18     {
19         //if boardAr at position [0][0] is equal to X and so is [0][1] then
20         //we check if Ai has already blocked [0][2] and if not than Ai blocks
21         //same logic applies to the rest of the ladder
22         thinking(boardAr);
23         boardAr[0][2] = token;
24         return true;
25     }
26     else if(boardAr[0][2] == 'X' && boardAr[0][1] == 'X' && boardAr[0][0] !=
27 '0') {
28         thinking(boardAr);
29         boardAr[0][0] = token;
30         return true;
31     }
32     else if(boardAr[1][0] == 'X' && boardAr[1][1] == 'X' && boardAr[1][2] !=
33 '0') {
34         thinking(boardAr);
35         boardAr[1][2] = token;
36         return true;
37     }
38     else if(boardAr[1][2] == 'X' && boardAr[1][1] == 'X' && boardAr[1][0] !=
39 '0') {
40         thinking(boardAr);
41         boardAr[1][0] = token;
42         return true;
43     }
44     else if(boardAr[2][0] == 'X' && boardAr[2][1] == 'X' && boardAr[2][2] !=
45 '0') {
46         thinking(boardAr);
47         boardAr[2][2] = token;
48         return true;
49     }
50     else if(boardAr[2][2] == 'X' && boardAr[2][1] == 'X' && boardAr[2][0] !=
51 '0') {
52         thinking(boardAr);
53         boardAr[2][0] = token;
54         return true;
55     }
56     else if(boardAr[0][0] == 'X' && boardAr[1][0] == 'X' && boardAr[2][0] !=
57 '0') {
58         thinking(boardAr);
59         boardAr[2][0] = token;
60         return true;
61     }
62     else if(boardAr[0][2] == 'X' && boardAr[1][2] == 'X' && boardAr[2][2] !=
63 '0') {
64         thinking(boardAr);
65         boardAr[2][2] = token;
66         return true;
67     }
68     else if(boardAr[0][0] == 'X' && boardAr[0][2] == 'X' && boardAr[1][0] !=
69 '0') {
70         thinking(boardAr);
71         boardAr[1][0] = token;
72         return true;
73     }
74     else if(boardAr[0][2] == 'X' && boardAr[0][0] == 'X' && boardAr[1][2] !=
75 '0') {
76         thinking(boardAr);
77         boardAr[1][2] = token;
78         return true;
79     }
80     else if(boardAr[1][0] == 'X' && boardAr[1][2] == 'X' && boardAr[2][0] !=
81 '0') {
82         thinking(boardAr);
83         boardAr[2][0] = token;
84         return true;
85     }
86     else if(boardAr[1][2] == 'X' && boardAr[1][0] == 'X' && boardAr[2][2] !=
87 '0') {
88         thinking(boardAr);
89         boardAr[2][2] = token;
90         return true;
91     }
92     else if(boardAr[2][0] == 'X' && boardAr[2][2] == 'X' && boardAr[1][0] !=
93 '0') {
94         thinking(boardAr);
95         boardAr[1][0] = token;
96         return true;
97     }
98     else if(boardAr[2][2] == 'X' && boardAr[2][0] == 'X' && boardAr[1][2] !=
99 '0') {
100        thinking(boardAr);
101        boardAr[1][2] = token;
102        return true;
103    }
104    return false;
105 }

```

```
52         return true;
53     }
54     else if(boardAr[2][0] == 'X' && boardAr[1][0] == 'X' && boardAr[0][0] !=
'0') {
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] !=
'0') {
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] !=
'0') {
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] !=
'0') {
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] !=
'0') {
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] !=
'0') {
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] !=
'0') {
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] !=
'0') {
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] !=
'0') {
95         thinking(boardAr);
96         boardAr[0][2] = token;
97         return true;
98     }
99     else
100         return false;
101 }
```

```

1  /*****
2  * Title: Thinking
3  * -----
4  * Function:
5  *   this function outputs thinking
6  * -----
7  * No Data Table
8  * -----
9  *****/
10
11 #include "main.h"
12
13 void thinking(char boardAr[][3])
14 {
15     for(size_t i {3}; i > 0; i--)
16     {
17         system("clear");
18         displayBoard(boardAr);
19         if(i == 3)
20             std::cout << "Thinking.\n";
21         else if(i == 2)
22             std::cout << "Thinking..\n";
23         else if(i == 1)
24             std::cout << "Thinking...\n";
25         usleep(150000);
26     }
27 }

```

```

1  /*****
2  * Title: normalWinConditionsAi
3  * -----
4  * Function:
5  * this function checks to see if there is a possible win condition for Ai
6  * and if there is then acts on it, normal only does some win possible
7  * win conditions
8  * -----
9  * No Data Table
10 * -----
11 *****/
12 #include "main.h"
13
14 bool normalWinConditionsAi(char boardAr[][3], const char token)
15 {
16     if(boardAr[0][0] == '0' && boardAr[0][1] == '0' && boardAr[0][2] != 'X') {
17         thinking(boardAr);
18         boardAr[0][2] = token;
19         return true;
20     }
21     else if(boardAr[0][2] == '0' && boardAr[0][1] == '0' && boardAr[0][0] !=
22 'X') {
23         thinking(boardAr);
24         boardAr[0][0] = token;
25         return true;
26     }
27     else if(boardAr[1][0] == '0' && boardAr[1][1] == '0' && boardAr[1][2] !=
28 'X') {
29         thinking(boardAr);
30         boardAr[1][2] = token;
31         return true;
32     }
33     else if(boardAr[1][2] == '0' && boardAr[1][1] == '0' && boardAr[1][0] !=
34 'X') {
35         thinking(boardAr);
36         boardAr[1][0] = token;
37         return true;
38     }
39     else if(boardAr[2][0] == '0' && boardAr[2][1] == '0' && boardAr[2][2] !=
40 'X') {
41         thinking(boardAr);
42         boardAr[2][2] = token;
43         return true;
44     }
45     else if(boardAr[2][2] == '0' && boardAr[2][1] == '0' && boardAr[2][0] !=
46 'X') {
47         thinking(boardAr);
48         boardAr[2][0] = token;
49         return true;
50     }
51     else if(boardAr[0][0] == '0' && boardAr[1][0] == '0' && boardAr[2][0] !=
52 'X') {
53         thinking(boardAr);

```

```

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

```

```

1  /*****
2  * Title: hardWinConditionsAi
3  * -----
4  * Function:
5  * This function with the addition of normalWinConditionsAi have every
6  * possible win condition hard coded to be able to make that move and win
7  * the game
8  * -----
9  * No Data Table
10 * -----
11 *****/
12
13 #include "main.h"
14
15 bool hardWinConditionsAi(char boardAr[][3], const char token)
16 {
17     if(boardAr[0][0] == '0' && boardAr[0][2] == '0' && boardAr[0][1] != 'X') {
18         thinking(boardAr);
19         boardAr[0][1] = token;
20         return true;
21     }
22     if(boardAr[1][0] == '0' && boardAr[1][2] == '0' && boardAr[1][1] != 'X') {
23         thinking(boardAr);
24         boardAr[1][1] = token;
25         return true;
26     }
27     if(boardAr[2][0] == '0' && boardAr[2][2] == '0' && boardAr[2][1] != 'X') {
28         thinking(boardAr);
29         boardAr[2][1] = token;
30         return true;
31     }
32     if(boardAr[0][0] == '0' && boardAr[2][0] == '0' && boardAr[1][0] != 'X') {
33         thinking(boardAr);
34         boardAr[1][0] = token;
35         return true;
36     }
37     if(boardAr[0][1] == '0' && boardAr[2][1] == '0' && boardAr[1][1] != 'X') {
38         thinking(boardAr);
39         boardAr[1][1] = token;
40         return true;
41     }
42     if(boardAr[0][2] == '0' && boardAr[2][2] == '0' && boardAr[1][2] != 'X') {
43         thinking(boardAr);
44         boardAr[1][2] = token;
45         return true;
46     }
47     if(boardAr[0][0] == '0' && boardAr[2][2] == '0' && boardAr[1][1] != 'X') {
48         thinking(boardAr);
49         boardAr[1][1] = token;
50         return true;
51     }
52     if(boardAr[0][2] == '0' && boardAr[2][0] == '0' && boardAr[1][1] != 'X') {
53         thinking(boardAr);
54         boardAr[1][1] = token;
55         return true;
56     }
57     else
58         return false;
59 }

```

```

1  /*****
2  * Title: hardAiBlocking
3  * -----
4  * Function:
5  *   Blocks any possible win conditions that player X has
6  * -----
7  * No Data Table
8  * -----
9  *****/
10
11 #include "main.h"
12
13 bool hardAiBlocking(char boardAr[][3], const char token)
14 {
15     if(boardAr[0][0] == 'X' && boardAr[0][2] == 'X' && boardAr[0][1] != '0') {
16         thinking(boardAr);
17         boardAr[0][1] = token;
18         return true;
19     }
20     if(boardAr[1][0] == 'X' && boardAr[1][2] == 'X' && boardAr[1][1] != '0') {
21         thinking(boardAr);
22         boardAr[1][1] = token;
23         return true;
24     }
25     if(boardAr[2][0] == 'X' && boardAr[2][2] == 'X' && boardAr[2][1] != '0') {
26         thinking(boardAr);
27         boardAr[2][1] = token;
28         return true;
29     }
30     if(boardAr[0][0] == 'X' && boardAr[2][0] == 'X' && boardAr[1][0] != '0') {
31         thinking(boardAr);
32         boardAr[1][0] = token;
33         return true;
34     }
35     if(boardAr[0][1] == 'X' && boardAr[2][1] == 'X' && boardAr[1][1] != '0') {
36         thinking(boardAr);
37         boardAr[1][1] = token;
38         return true;
39     }
40     if(boardAr[0][2] == 'X' && boardAr[2][2] == 'X' && boardAr[1][2] != '0') {
41         thinking(boardAr);
42         boardAr[1][2] = token;
43         return true;
44     }
45     if(boardAr[0][0] == 'X' && boardAr[2][2] == 'X' && boardAr[1][1] != '0') {
46         thinking(boardAr);
47         boardAr[1][1] = token;
48         return true;
49     }
50     if(boardAr[0][2] == 'X' && boardAr[2][0] == 'X' && boardAr[1][1] != '0') {
51         thinking(boardAr);
52         boardAr[1][1] = token;
53         return true;
54     }
55     else
56         return false;
57 }

```