

Final Project

<ULTIMATE TIC-TAC-TOE>

Course

CSC-5 Fall 2013

Section

47982

Due Date

December 9th, 2013

Author

Oscar Martinez

Table of Contents

1 Introduction.....	3
2 Game Play and Rules	3
3 Development Summary	3
4 Flowchart	4
5 Sample Inputs/Outputs	5
5.1 Tied Game	5
5.2 Player Won	6
5.3 Invalid Move.....	6
5.4 Option to play again	7
6 Pseudo Code	7
7 Variables Used	8
8 Concepts Used	8
9 Functions.....	9
10 References.....	8
11 Program Listing	9

1 Introduction

Project title: ULTIMATE TIC-TAC-TOE

For the programming project, I decided to go with TIC-TAC-TOE. I think that TIC-TAC-TOE is one of the first games that we learn as kids. This game is a great game to play when one has some free and all one needs is a computer or even just a simple piece of paper. I started with the idea of doing TIC-TAC-TOE for my project and got started writing the code thinking that it would be worthy of a programming game, without taking into account that this project has to be further expanded for the final project. When Dr. Lehr mentioned in class that TIC-TAC-TOE would be difficult to expand, it was already too late to start on something else. I should have probably consulted my idea with Dr. Lehr before getting too far into it. Hopefully there is a way to further expand this game to be worthy of a final project.

At first I did not know what to do for the final project. Besides putting code into functions, there isn't really much one can do to make TIC-TAC-TOE more difficult. I did some searching online and found a website that talked about ULTIMATE TIC-TAC-TOE. From the pictures the website had, the concept seem rather interesting. What ULTIMATE TIC-TAC-TOE is is basically a game within a game. In this game I managed to utilize most of the concepts that we had covered in the second half of the semester and I hope that it satisfies the expectations of a final project.

2 Game Play and Rules

The game consists of a board with nine empty boxes where one of two players who are either an X or an O will take turns placing their letter on a box. Whichever player is able to get three of his/her letters in a row wins. The three letters have to be together either in a row column or diagonally. Should every box have a letter and there is no winner the game will be tied.

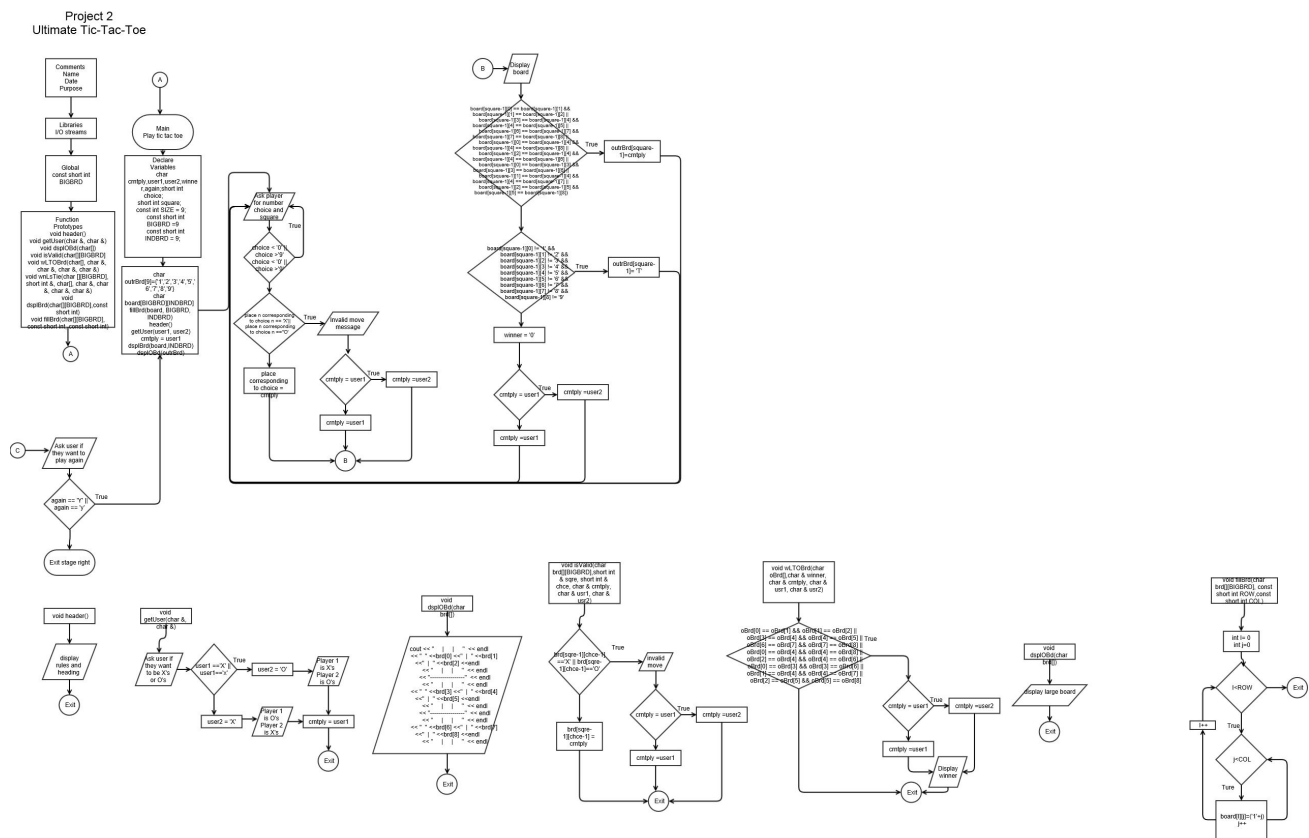
The rules for ULTIMATE TIC-TAC-TOE are basically the same as for regular TIC-TAC-TOE. The players still take turns putting either an X or an O on the square. The board however, consists of TIC-TAC-TOE boards in each of the squares of the bigger TIC-TAC-TOE board. Whichever player goes first can go anywhere in the 81 squares. Player one chooses a board and then a square within the board. Player two has to make a move in the board that corresponds to the square that the player one chose. The players take turns and the object is to win three boards to win the whole game. Should the players tie in a board, the board does not count for either player and the next player has free range to go wherever they want. First player that wins three boards wins the game.

3 Development Summary

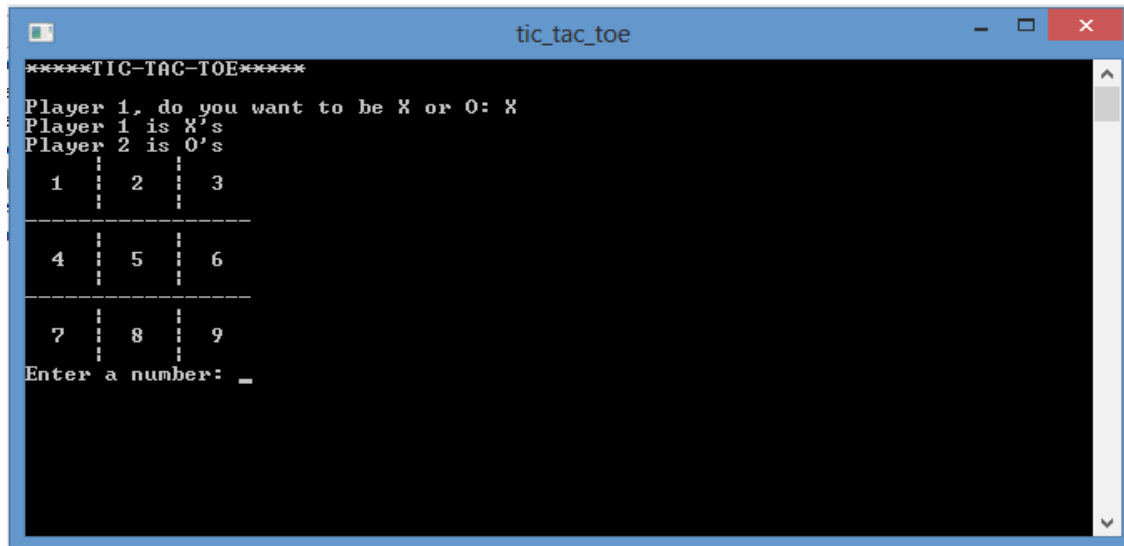
Lines of code	206
Comment Lines	93
Blank Lines(White Space)	58
Variables Used	7
Total Project Size	338
Functions used	7

This project was created using the Netbeans IDE and the folwchart was created using Gliffy.

4 Flowchart



5 Sample Inputs/Outputs

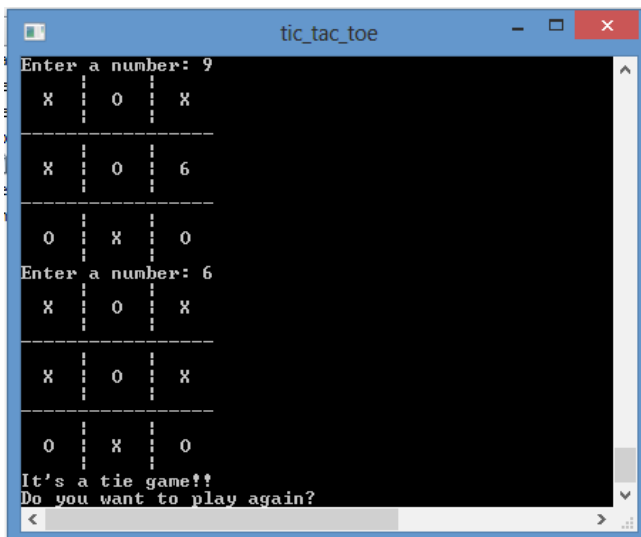


```
*****TIC-TAC-TOE*****
Player 1, do you want to be X or O: X
Player 1 is X's
Player 2 is O's

 1 | 2 | 3
--|---
 4 | 5 | 6
--|---
 7 | 8 | 9
Enter a number: _
```

This screen shows the start of the game

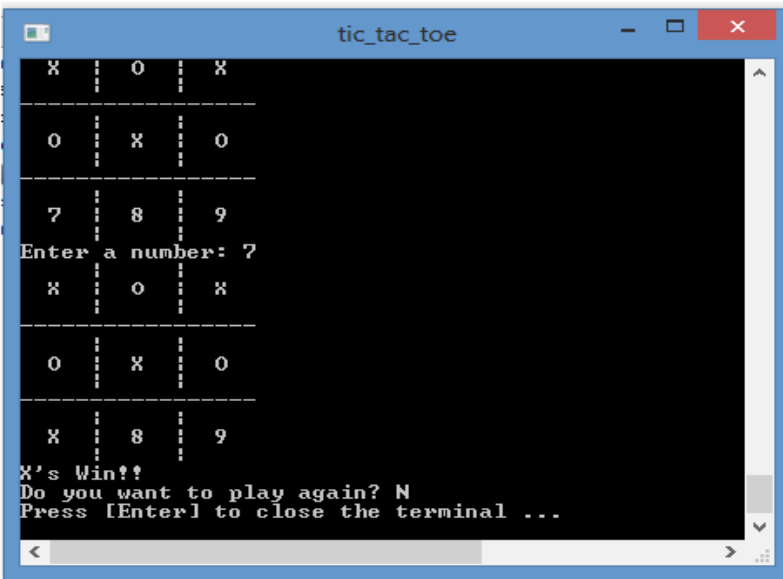
5.1 Tied Game



```
Enter a number: 9
 X | O | X
--|---
 X | O | 6
--|---
 O | X | O
Enter a number: 6
 X | O | X
--|---
 X | O | X
--|---
 O | X | O
It's a tie game!!
Do you want to play again?
```

This screen demonstrates a tied game and asks the user if they would like to play again.

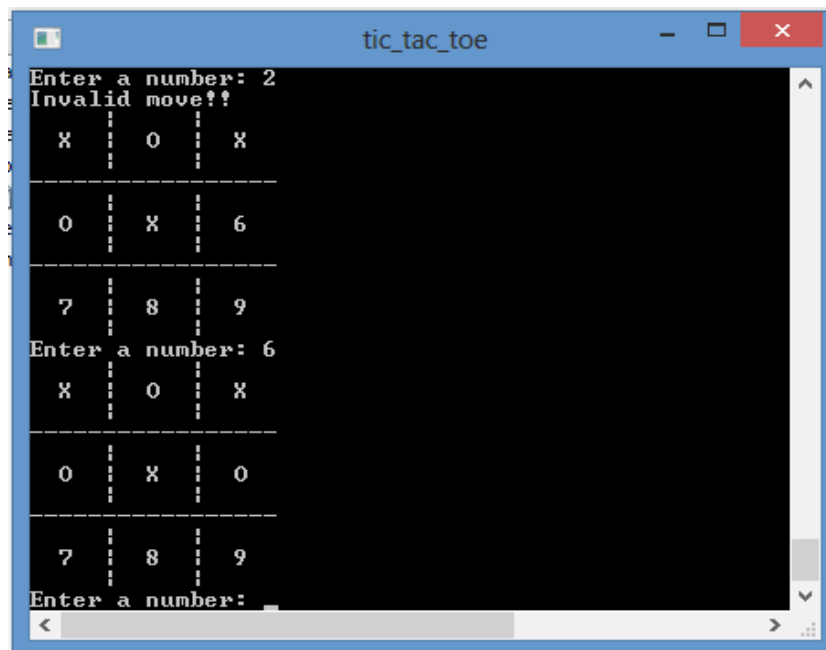
5.2 Player Won



```
tic_tac_toe
X | O | X
---|---|---
O | X | O
---|---|---
7 | 8 | 9
Enter a number: 7
X | O | X
---|---|---
O | X | O
---|---|---
X | 8 | 9
X's Win!!
Do you want to play again? N
Press [Enter] to close the terminal ...
```

In this case the player that is X's wins and he/she is asked if they would like to play again.

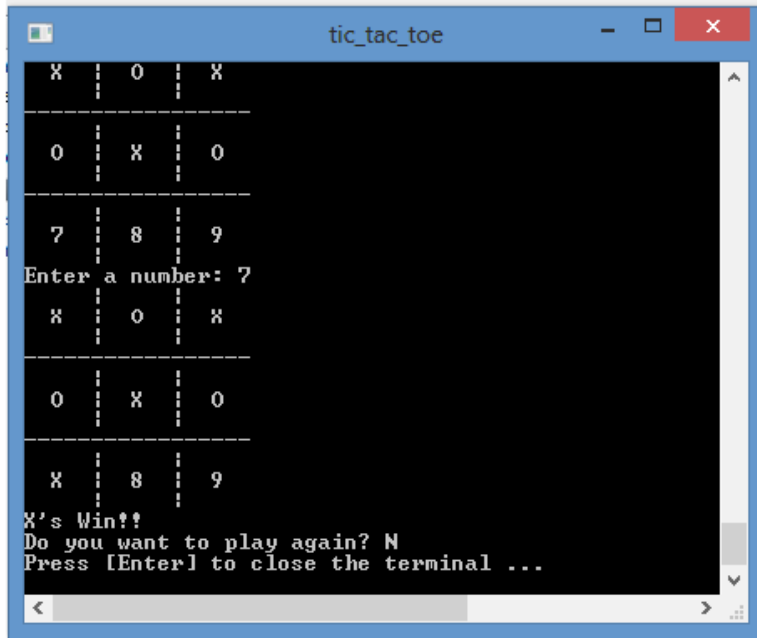
5.3 Invalid Move



```
tic_tac_toe
Enter a number: 2
Invalid move!!
X | O | X
---|---|---
O | X | 6
---|---|---
7 | 8 | 9
Enter a number: 6
X | O | X
---|---|---
O | X | O
---|---|---
7 | 8 | 9
Enter a number: 
```

Should the player input a number that is already taken he/she will be asked to input another number.

5.4 Option to play again



Once the game is over, the user is asked if they would like to play again. If so, the game is restarted and if not, then the program exits.

6 Pseudo Code

```
//Declare variables
//Assign numbers to each box
//Heading
//Ask the user if they want to be X's or O's
//Sets user1 to current player
//Display board
//Prompts user for a move and validates
//Keeps switching between players ans asking for a play while there is no winner
//Checks that the move made by the user is one of the choices on the grid
//Changes the number on the grid to the current player's letter
** The following is repeated for choices 1-9
//Checks to see if the move made by the current user is valid
//meaning there is no X or O
//If the move is invalid, the current player is changed
//so that the same player can go again
//If the move is valid, the number chosen by the player changes to X or O
**
```

```
//Displays board
//Checks if current player won ( row, column, diagonal)
// Checks for tied game
//If no winner and not a tied game, switches player and asks for another move
//Checks if player won the the whole game
//Asks player if they want to play again if there's a winner or it's tied
```

7 Variables Used

Variable Name	Purpose
choice	Player's move
crntply	Current player (X or O)
user1 & user2	Keeps track of player's game piece (X or O)
winner	Holds the status of winner
again	Choice of whether a player wants to play again
square	The board where player wants to make next move
SIZE	Size of individual boards
BIGBRD	Size of row
INDBRD	Size of column
outrBrd[]	Array of individual boards
board[][]	2D array of big board

8 Concepts Used

Chapter	Topics used
2	Cout object, #include directive, char data type, variable assignments and initialization, comments, programming style.
3	cin object
4	Relational operators, the if statement, the if/else statement, the if/else if statement, validating user input, switch statement, logical operators, comparing characters and strings
5	Do-while loop, sentinels

9 Functions

void header()	Displays rules of the game
void getUser(char &user1,char &user2)	Prompts user for choice of X or O
void dsplOBd(char brd[])	Displays the large board with embedded boards
void isValid(char brd[][BIGBRD],short int & sqre, short int & chce, char & crntply, char & usr1, char & usr2)	Validates the move made by player and if move is valid, it changes the square to the player's letter
void wLTOBrd(char oBrd[],char & winner, char & crntply, char & usr1, char & usr2)	Checks if a player has won three boards and displays the winner
void wnLsTie(char brd[][BIGBRD],int sqre, char oBrd[],char & winner, char & crntply, char & usr1, char & usr2)	Checks the individual boards to see if the current player won that board
void dsplBrd(char brd[][BIGBRD], const short int COL)	Displays the big board
void fillBrd(char brd[][BIGBRD], const short int ROW,const short int COL)	Fills the boards with numbers 1-9

10 References

Gaddis, Tony. *Starting out with C : From Control Structures through Objects*. 6th ed. Boston: Pearson Addison-Wesley, 2012. Print.

Savitch, Walter J., and Kenrick Mock. *Problem Solving with C*. 8th ed. Boston: Pearson Addison Wesley, 2012. Print.

Write ups of previous projects

Concept of Ultimate Tic Tac Toe

<http://mathwithbaddrawings.com/2013/06/16/ultimate-tic-tac-toe/>

11 Program Listing

```
/*
```

```
* File:  main.cpp
```

```
* Author: Oscar Martinez
```

```
* Created on October 10, 2013, 7:37 PM
```

```
* Purpose: Tic-Tac-Toe Final Project
```

```
*/
```

```
//Libraries
```

```
#include <iostream>
```

```
using namespace std;
```

```
//Global constants
```

```
const short int BIGBRD = 9;
```

```
//Function prototypes
```

```
void header();           //Outputs rules and header
```

```
void getUser(char &, char &); //Gets user
```

```
void dsplOBd(char[]);     //Displays big board
```

```
void isValid(char[][BIGBRD], short int &, short int &, char &, char &, char &); //Validates the move
```

```
void wLTObd(char[], char &, char &, char &, char &); //Checks is someone won the overall  
game
```

```
void wnLsTie(char[][BIGBRD], short int &, char[], char &, char &, char &, char &); //Checks if  
someone won individual game
```

```
void dsplBrd(char[][BIGBRD], const short int); //Displays big board
```

```
void fillBrd(char[][BIGBRD], const short int, const short int); //Fills big board with numbers
```

```
//Execution begins here
```

```
int main(int argc, char** argv) {
```

```
    //Declare variables here
```

```
    char crntply,           //Current Player
```

```
        user1, user2,      //Player one and player two
```

```
        winner,           //Winner of game
```

```
        again;            //Choice to play again
```

```
    short int choice;       //choice within individual board
```

```
    short int square;       //Choice of square
```

```
    const int SIZE = 9;     //Size of individual boards
```

```
    const short int BIGBRD = 9; //Number of squares in big board
```

```
    const short int INDBRD = 9; //Number of squares in each individual board
```

```
do{
```

```
    //Create the board
```

```
    char outrBrd[9]={'1','2','3','4','5','6','7','8','9'};
```

```

//Create board containing individual boards
char board[BIGBRD][INDBRD];

//fill the board
fillBrd(board, BIGBRD, INDBRD);

//Heading
header();

//Ask user if they want to be X's or O's
getUser(user1, user2);

//Sets user1 to the current player
crntply = user1;

//Display the board containing individual boards
dsplBrd(board,INDBRD);

//Display Outer Board
dsplOBd(outrBrd);

//Prompts user for a move and validates
//Keeps switching between players and asking for a play while there is no winner
do{
    //Checks that the move made by user is one of the choices on the grid
    do{
        cout << "Chose a board: ";
        cin >> square;
        cout << "Enter a number: ";
        cin >> choice;
    }while(choice < 0 || choice > 9);

    //Changes the number on the grid to the current player's letter

```

```

//Checks to see if the move made by current user is valid
//meaning there is no X or O
isValid(board, square, choice, crntply, user1, user2);

//Display board
dsplBrd(board,BIGBRD);

//Checks to see if current player won square
//Checks if the rows, columns or diagonals are the same
if(board[square-1][0] == board[square-1][1] && board[square-1][1] == board[square-1][2] ||
   board[square-1][3] == board[square-1][4] && board[square-1][4] == board[square-1][5] ||
   board[square-1][6] == board[square-1][7] && board[square-1][7] == board[square-1][8] ||
   board[square-1][0] == board[square-1][4] && board[square-1][4] == board[square-1][8] ||
   board[square-1][2] == board[square-1][4] && board[square-1][4] == board[square-1][6] ||
   board[square-1][0] == board[square-1][3] && board[square-1][3] == board[square-1][6] ||
   board[square-1][1] == board[square-1][4] && board[square-1][4] == board[square-1][7] ||
   board[square-1][2] == board[square-1][5] && board[square-1][5] == board[square-1][8]){
//If a row, diagonal, or column is the same the player that
//just went is the winner
   outrBrd[square-1]=crntply;
   cout<< crntply <<"s win square " <<(square) <<endl;
//Switches players
   if(crntply==user1) crntply=user2;
   else crntply= user1;
}
//If all the boxes have either an O or X, and no one won,
//the game is tied
else if( board[square-1][0] != '1' &&
        board[square-1][1] != '2' &&
        board[square-1][2] != '3' &&
        board[square-1][3] != '4' &&
        board[square-1][4] != '5' &&
        board[square-1][5] != '6' &&

```

```

        board[square-1][6] != '7' &&
        board[square-1][7] != '8' &&
        board[square-1][8] != '9'){
//Displays tie game message
cout << "It's a tie game for square " << (square) << " !!" <<endl;
//Sets the square in the big board to T to signal a tie
outrBrd[square-1]='T';
//Switches Player
if(crntply==user1) crntply=user2;
else crntply= user1;
}
//There is no winner
else{
    //Sets winner to 0 so it can keep asking for moves
    winner = '0';
    //Switches players
    if(crntply == user1){
        crntply = user2;
    }
    else{
        crntply = user1;
    }
}
//Checks to see if the current player won the big board
//Checks if the rows, columns or diagonals are the same
//Checks if there is a tie
wLTObRd(outrBrd,winner, crntply, user1, user2);
dspIOBd(outrBrd);
}while(winner == '0');
//Prompts the user if they want to play again
cout << "Do you want to play again? ";
cin >> again;
}while(again == 'Y' || again == 'y');
return 0;

```

```

}
void header(){
    //Displays the header and the rules
    cout << "*****ULTIMATE TIC-TAC-TOE*****" <<endl <<endl;
    cout << "This game of TIC-TAC-TOE involves a game within a game."<<endl <<"The object is to "
        <<"win three boards to win the whole game."<<endl <<"Wherever the last player went in the
individual "
        <<"board is where"<<endl <<"the next player goes in the big board. Should an individual board
be tied, "
        <<endl <<"it doesn't count for either player. " <<endl <<endl;
}
void getUser(char &user1,char &user2){
    //Ask the user if they want to be X's or O's
    cout << "Player 1, do you want to be X or O: ";
    cin >> user1;

    //Displays each player with their corresponding letter (X or O)
    if(user1 == 'X' || user1 == 'x'){
        user1='X';
        user2 = 'O';
        cout << "Player 1 is X's" <<endl
            << "Player 2 is O's" <<endl;
    }
    else if( user1=='o' || user1=='O'){
        user1=='O';
        user2 = 'X';
        cout << "Player 1 is O's" <<endl
            << "Player 2 is X's" <<endl;
    }
}

void dspLOBd(char brd[]){
    //Displays the big board
    cout << "   |   |   " << endl
        << " " <<brd[0] <<" | " <<brd[1] <<" | " <<brd[2] <<endl

```

```

    << "    |    |    " << endl
    << "-----" << endl
    << "    |    |    " << endl
    << " " << brd[3] << " | " << brd[4] << " | " << brd[5] << endl
    << "    |    |    " << endl
    << "-----" << endl
    << "    |    |    " << endl
    << " " << brd[6] << " | " << brd[7] << " | " << brd[8] << endl
    << "    |    |    " << endl;
}

```

```

void isValid(char brd[][BIGBRD], short int & sqre, short int & chce, char & crntply, char & usr1, char
& usr2){

```

```

    //Changes the number on the grid to the current player's letter

```

```

    //Checks to see if the move made by current user is valid

```

```

    //meaning there is no X or O

```

```

    if (brd[sqre-1][chce-1] == 'X' || brd[sqre-1][chce-1] == 'O'){

```

```

        cout << "Invalid move!!" << endl;

```

```

        //If the move is invalid, the current user is changed

```

```

        //so that the same player can go again

```

```

        if (crntply == usr1) crntply = usr2;

```

```

        else crntply = usr1;

```

```

    }

```

```

    //If the move is valid, the number chosen by player changes

```

```

    //to X or O

```

```

    else brd[sqre-1][chce-1] = crntply;

```

```

}

```

```

void wLTObRd(char oBrd[], char & winner, char & crntply, char & usr1, char & usr2){

```

```

    //Checks to see if the current player won

```

```

    //Checks if the rows, columns or diagonals are the same

```

```

    if (oBrd[0] == oBrd[1] && oBrd[1] == oBrd[2] ||

```

```

        oBrd[3] == oBrd[4] && oBrd[4] == oBrd[5] ||

```

```

oBrd[6] == oBrd[7] && oBrd[7] == oBrd[8] ||
oBrd[0] == oBrd[4] && oBrd[4] == oBrd[8] ||
oBrd[2] == oBrd[4] && oBrd[4] == oBrd[6] ||
oBrd[0] == oBrd[3] && oBrd[3] == oBrd[6] ||
oBrd[1] == oBrd[4] && oBrd[4] == oBrd[7] ||
oBrd[2] == oBrd[5] && oBrd[5] == oBrd[8]){
//If a row, diagonal, or column is the same the player that
//just went is the winner
if(crntply==usr1) crntply=usr2;
else crntply= usr1;
winner = crntply;

//Displays the winner
cout << winner <<"s Win!!" <<endl;
}
//If all the boxes have either an O or X, and no one won,
//the game is tied
else if( oBrd[0] != '1' &&
        oBrd[1] != '2' &&
        oBrd[2] != '3' &&
        oBrd[3] != '4' &&
        oBrd[4] != '5' &&
        oBrd[5] != '6' &&
        oBrd[6] != '7' &&
        oBrd[7] != '8' &&
        oBrd[8] != '9'){
//Displays tie game message
cout << "It's a tie game!!" <<endl;
//Sets winner to 1 so that it can exit the loop and stop asking for moves
winner = '1';
}
}

```



```
void wnLsTie(char brd[][BIGBRD],int sqre, char oBrd[],char & winner, char & crntply, char & usr1, char & usr2){
```

```
    //Checks to see if the current player won
```

```
    //Checks if the rows, columns or diagonals are the same
```

```
    if(brd[sqre][0] == brd[sqre][1] && brd[sqre][1] == brd[sqre][2] ||
        brd[sqre][3] == brd[sqre][4] && brd[sqre][4] == brd[sqre][5] ||
        brd[sqre][6] == brd[sqre][7] && brd[sqre][7] == brd[sqre][8] ||
        brd[sqre][0] == brd[sqre][4] && brd[sqre][4] == brd[sqre][8] ||
        brd[sqre][2] == brd[sqre][4] && brd[sqre][4] == brd[sqre][6] ||
        brd[sqre][0] == brd[sqre][3] && brd[sqre][3] == brd[sqre][6] ||
        brd[sqre][1] == brd[sqre][4] && brd[sqre][4] == brd[sqre][7] ||
        brd[sqre][2] == brd[sqre][5] && brd[sqre][5] == brd[sqre][8]){
```

```
        //If a row, diagonal, or column is the same the player that
```

```
        //just went is the winner
```

```
        oBrd[sqre-1]=crntply;
```

```
        cout<< crntply <<"s win square " <<(sqre-1);
```

```
    }
```

```
    //If all the boxes have either an O or X, and no one won,
```

```
    //the game is tied
```

```
    else if( brd[sqre][0] != '1' &&
```

```
        brd[sqre][1] != '2' &&
```

```
        brd[sqre][2] != '3' &&
```

```
        brd[sqre][3] != '4' &&
```

```
        brd[sqre][4] != '5' &&
```

```
        brd[sqre][5] != '6' &&
```

```
        brd[sqre][6] != '7' &&
```

```
        brd[sqre][7] != '8' &&
```

```
        brd[sqre][8] != '9'){
```

```
        //Displays tie game message
```

```
        cout << "It's a tie game for square " << (sqre-1) <<" !!!" <<endl;
```

```
        //Sets winner to 1 so that it can exit the loop and stop asking for moves
```

```
        oBrd[sqre-1]='T';
```

```
    }
```

```
    //There is no winner
```

```

else{
    //Sets winner to 0 so it can keep asking for moves
    winner = '0';
    //Switches players
    if(crntply == usr1){
        crntply = usr2;
    }
    else{
        crntply = usr1;
    }
}
}

```

```

void dsplBrd(char brd[][BIGBRD], const short int COL){

```

```

    //Displays the big board, which is a tic-tac-toe board but in each square there is another tic-tac-toe
    game inbedded.

```

```

    cout<< "   |   |   |" << "|   |   |" << "|   |   |" << endl
        << " " << brd[0][0] << " | " << brd[0][1] << " | " << brd[0][2] << " || " << brd[1][0] << " | "
<< brd[1][1] << " | " << brd[1][2] << " || " << brd[2][0] << " | " << brd[2][1] << " | " << brd[2][2]
<< endl
        << "   |   |   |" << "|   |   |" << "|   |   |" << endl
        << "-----|" << "|-----|" << "|-----" << endl
        << "   |   |   |" << "|   |   |" << "|   |   |" << endl
        << " " << brd[0][3] << " | " << brd[0][4] << " | " << brd[0][5] << " || " << brd[1][3] << " | "
<< brd[1][4] << " | " << brd[1][5] << " || " << brd[2][3] << " | " << brd[2][4] << " | " << brd[2][5]
<< endl
        << "   |   |   |" << "|   |   |" << "|   |   |" << endl
        << "-----|" << "|-----|" << "|-----" << endl
        << "   |   |   |" << "|   |   |" << "|   |   |" << endl
        << " " << brd[0][6] << " | " << brd[0][7] << " | " << brd[0][8] << " || " << brd[1][6] << " | "
<< brd[1][7] << " | " << brd[1][8] << " || " << brd[2][6] << " | " << brd[2][7] << " | " << brd[2][8]
<< endl
        << "   |   |   |" << "|   |   |" << "|   |   |" << endl;
    cout << "===== " << endl;
    cout<< "   |   |   |" << "|   |   |" << "|   |   |" << endl
        << " " << brd[3][0] << " | " << brd[3][1] << " | " << brd[3][2] << " || " << brd[4][0] << " | "
<< brd[4][1] << " | " << brd[4][2] << " || " << brd[5][0] << " | " << brd[5][1] << " | " << brd[5][2]

```

```

<<endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << "-----|" << "-----|" << "-----|" << endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << " " <<brd[3][3] <<" | " <<brd[3][4] <<" | " <<brd[3][5] << " || " <<brd[4][3] <<" | "
<<brd[4][4] <<" | " <<brd[4][5] << " || " <<brd[5][3] <<" | " <<brd[5][4] <<" | " <<brd[5][5]
<<endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << "-----|" << "-----|" << "-----|" << endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << " " <<brd[3][6] <<" | " <<brd[3][7] <<" | " <<brd[3][8] << " || " <<brd[4][6] <<" | "
<<brd[4][7] <<" | " <<brd[4][8] << " || " <<brd[5][6] <<" | " <<brd[5][7] <<" | " <<brd[5][8]
<<endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl;
    cout <<"===== "<<endl;
    cout<< "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << " " <<brd[6][0] <<" | " <<brd[6][1] <<" | " <<brd[6][2] << " || " <<brd[7][0] <<" | "
<<brd[7][1] <<" | " <<brd[7][2] << " || " <<brd[8][0] <<" | " <<brd[8][1] <<" | " <<brd[8][2]
<<endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << "-----|" << "-----|" << "-----|" << endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << " " <<brd[6][3] <<" | " <<brd[6][4] <<" | " <<brd[6][5] << " || " <<brd[7][3] <<" | "
<<brd[7][4] <<" | " <<brd[7][5] << " || " <<brd[8][3] <<" | " <<brd[8][4] <<" | " <<brd[8][5]
<<endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << "-----|" << "-----|" << "-----|" << endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl
    << " " <<brd[6][6] <<" | " <<brd[6][7] <<" | " <<brd[6][8] << " || " <<brd[7][6] <<" | "
<<brd[7][7] <<" | " <<brd[7][8] << " || " <<brd[8][6] <<" | " <<brd[8][7] <<" | " <<brd[8][8]
<<endl
    << "    |    |    |" << "    |    |    |" << "    |    |    |" << endl;

}

```

```

void fillBrd(char brd[][BIGBRD], const short int ROW,const short int COL) {
    //Fills each individual board with numbers 1-9

```

```
for(int i = 0; i<ROW; i++){  
    for(int j =0; j<COL; j++){  
        brd[i][j]='1'+j);  
    }  
}  
}
```