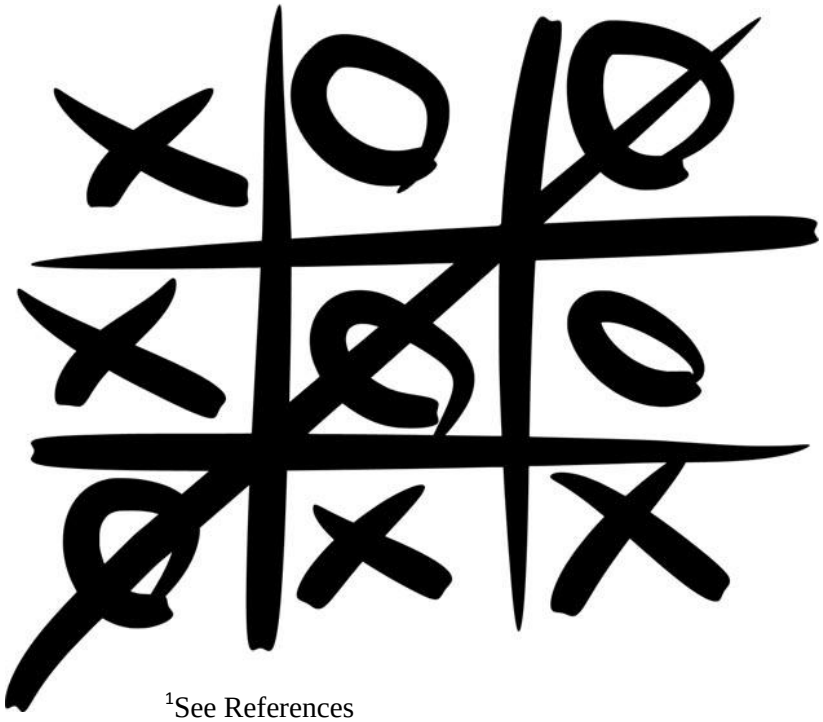
# PROJECT 2
## TIC TAC TOE



[1]See References

Anh Vu
CSC 5 – 46023
July 31st, 2014

## TABLE OF CONTENTS

## INTRODUCTION

Tic-Tac-Toe is a game in which two players, represented as either X or O, mark a 3 x 3 grid until one player wins. The objective of the game is to mark 3 spaces that are horizontally, vertically, or diagonally adjacent to one another before the other player does.

**Rules –**

- Each player will take turns marking the grid. No player will mark the grid more than once at a time.
- You must only mark the empty spaces in the grid.
- Many times you will find that neither player wins. This game is considered a draw, or a cat's game.

**Instructions –**

- The program will randomly decide which player ('X' or 'O') will play first
- Each space in the 3 x 3 grid will be represented by a number. Please select the numbered space in which you would like to place your mark. Your mark will be placed on the grid after each time you input.
- The game will continue to run until 1) one person wins, or 2) all spaces are occupied and no one wins.
- If you wish to see your total scores, see the text file names "scores"
- Enjoy the game!

## DESIGN DETAILS

**Approach -**

In order to successfully code Tic-Tac-Toe, I first played a game and broke the game down in small steps in order to fully understand the sub components of the game and to ultimately implement them in a code. Below are the steps I initially drafted:

1. Draw and output a 3x3 board.
2. Designate each player as either 'X' or 'O'
3. Allow users to take turns and to mark the board with their respective marks.
4. Conditions for the game to end
5. Ways which players can win
6. Keeping scores and outputting them so user can see

**Overview -**

At the end of my project, I had developed a game that had encompassed all of the steps that I had originally thought of. (Step 1) My program outputs a board each time a player makes a move. The board I created consists of a set of variables that represents each space. An example of my board is below:

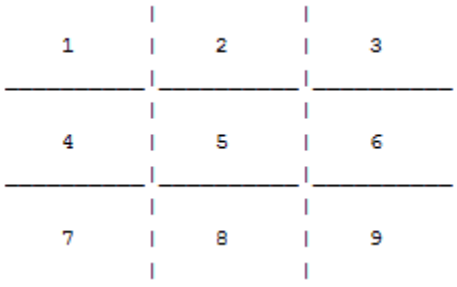



Figure 1. My tic-tac-toe board. In order to make a move, each player is required to input the number where he or she wants to place his or her mark.

If the space is already taken up, the program prompts the user to input another number. (Step 2) My program also designates Player 1 as 'X' and Player 2 as 'O' and randomly chooses which

player will be able to play first. Then, the program switches between each player from one move to the next. (Step 3) When the player chooses a number to place his mark, the number in the grid will be substituted with the mark. An example is shown below:

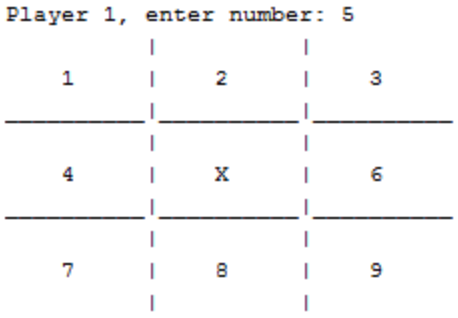



Figure 2. When player 1 (X) chooses number 5, that specific space is then replaced with player 1's mark.

(Step 4) Of course, when playing any game there are conditions to signal the end of the game. The game can end in nine different ways. Three ways the game can is if there are the same marks adjacent to one another in each of the three horizontal rows of the board (e.g. spaces 1, 2, and 3). Another three ways the game can end is the same as the aforementioned condition, but it applies to each of the three vertical columns. Two ways that will end the game is if the same marks are adjacent to one another diagonally. The last way the game can end is if all the spaces are taken up. In this case, neither player wins and the game is considered a draw. Below shows an example of an output to signal the end of the game. Also note that I prompt the user to decide whether he or she wants to play again.

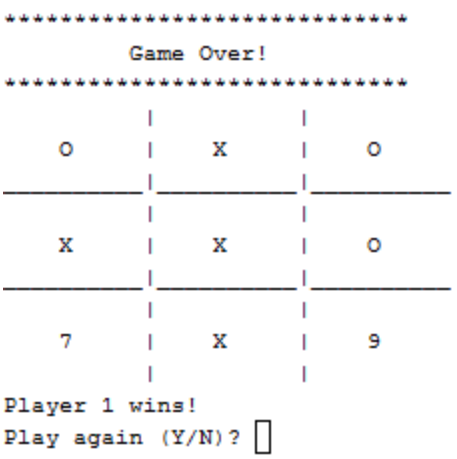



Figure 3. Player 1 wins because he/she placed 3 marks adjacent to one another vertically before the other player could. The program also asks the user if he or she wants to play again.

(Step 5) As you can see in *figure 3*, the program is able to determine who wins. The condition that determines who wins is which player is able to end the game first. (Step 6) When each game ends, the scores for each player are subsequently added depending on which player wins. Scores are tallied and output on a text file titled "scores." Below is a sample of a the text file:

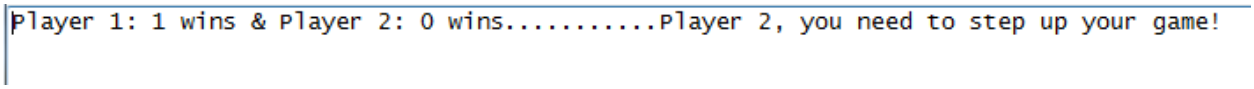



Figure 4. Player 1 has one win while Player 2 has none.

The details of my program will be further discussed throughout my report.

## RESEARCH

In order to code my game, I had to research and study the following topics that I found difficult to understand:

1. **Arrays -**

   I used arrays so I could hold the variables that would represent each space on the board. When the user inputs where they want their mark, that variable in the array would be replaced with the user's respective mark. I mostly used arrays in order to hold a lot of variables on my tic-tac-toe board; this made it very easy for me to reassign each space with a mark throughout the game. Initially, I had used a one dimensional array to hold my variables. In order to encompass my understanding of 2D arrays into the project, I converted my array from a 1D to a 2D array, with 3 columns and 3 rows.

2. **Functions -**

   Without functions, my code would have been unnecessarily long. I would have to repeat my code to output a grid every time a player makes a move. The same applies for

determining the winner or whether the game is over. Instead of repeating the same lines of code over and over again, declaring a function and defining them would be much easier. Instead of repeating lines of code, I could call functions that would perform these codes. It greatly simplified the layout of my code and made it much easier to read and understand. I have separate functions for printing out the grid, determining whether the game is over, determining the winner, and outputting the right marks on each space.

3. **Bubble Sort -**

I had a really hard time thinking of a way to incorporate sorting into my project. Finally, I decided to sort all the elements in my board array after the game was over. When sorted, the grid should do a little dance and sort itself out. When done sorting, it should display the remaining numbers in ascending order, then the O's and then the X's, since the ASCII code for capital O is less in value than X.

## FLOWCHART

My flowchart was too large to place in my report, so please refer either to the JPEG files in my GitHub account, or the JPEG files titled "Project 2 Part 1" and "Project 2 Part 2" located in my Project 2 folder.

## VARIABLE LIST

| Variables & System Libraries | Purpose |
|---|---|
| `#include <iostream>` | Used for cout and cin |
| `#include <ctime>` | Necessary for setting random seed |
| `#include <cstdlib>` | Generating random number |
| `#include <fstream>` | File output |
| `#include <iomanip>` | Used to format my board, etc |
| `const int ROW=3` | Number of rows in grid |
| `const int COL=3` | Number of columns in grid |

| | |
|---|---|
| `char grid[ROW][COL]` | 2D array used to hold variables for spaces on the tic tac toe board. Has 3 columns and 3 rows |
| `bool p1` | Decides who's turn it is and which player is which mark |
| `unsigned short score1=0` | Used to count score for player 1 |
| `unsigned short score2=0` | Count score for player 2 |
| `char again` | Option that allows user to play game again. |
| `bool over` | Determine whether the game is over |
| `Unsigned short choice` | Where user wants to place his or her mark. Also used to allows user to choose from menu |
| `Short win` | Determine which player wins. |
| `Int i, j` | These variables were used to loop through the 2d array various times throughout my code |
| `Int num` | I used this number to print out numbers when I reset my board after each game |
| `Char x, y, temp` | Used to swap values during bubble sort |
| `Char ch, row, column` | Used to take user's choice, then converts to coordinates on the board |

## FUNCTION LIST

| Function | Purpose |
|---|---|
| `void welcome()` | Displays welcome message and randomly chooses who plays first |
| `void displayGrid(char[][COL],const int)` | Prints out Tic-Tac-Toe board |
| `void takeTurn(char[][COL],const int,bool)` | Places mark on board, determines who's turn it is |
| `bool gameOver(char[][COL],const int)` | Checks conditions to determine if game is over |
| `short winner(char[][COL],const int)` | Determines winner |
| `void menu()` | Displays menu before game starts |
| `void reset (char[][COL],const int)` | Resets board after game |
| `void file (int,const int)` | Outputs scores to file |

| | |
|---|---|
| `void sort(char[][COL],const int)` | Sorts board after each game |
| `void swap(char&,char&)` | Swaps values for sort |
| `void stall()` | Stalls program so we can see board sorting itself out |
| `void conversion(char,int&,int&)` | Converts user's number choice to coordinate to place mark |

## TOPICS COVERED

| Topic | Examples |
|---|---|
| Primitive Data Types | ***Boolean*** (Line 40)<br>***Char*** (Line 44)<br>***Int*** (Line 60)<br>***Short*** (Line 41)<br>*I did not use floats, I couldn't think of a way to incorporate a necessary float into my program. |
| System Level Libraries | ***Iostream***<br>***Cstdlib***<br>***Ctime***<br>***Fstream*** |
| Operators | Lines 207-232<br>105<br>(&&, \|\|, ==, ++,etc) |
| Conditionals | ***Do While*** (Line 69-82)<br>***If else*** (Line 77)<br>***Switch*** (Line 395)<br>***For*** (Line 245) |
| Menu | Line 289 |
| 2D Arrays | Line 39 |
| Functions | Line 21 |
| Bubble Sorting | 363 |
| Reading and Writing to a file | Reading from file (Line 57)<br>Writing to file (Line 339) |
| Generating Random Number | Lines 155, 158 |
| Default Parameters | Line 337 |

## DIFFERENCES BETWEEN PROJECT 1& 2

1. In project 1, I changed my grid array from a 1D to a 2D array. In order to do this I had to declare a constant variable named COL so I would be able to pass my array through functions.
2. I added more functions in order to main my main function easier to read. I also had to include an extra function in order to convert the user's choice to coordinates on the 2D array
3. I added more variables. You can see all of my functions and variables in the lists above this section.
4. I included a bubble sort to sort the elements in the array after the game ends. I included a default parameter as well.

## PSEUDOCODE

```
/*
 * File:   main.cpp
 * Author: Anh Vu
 *
 * Created on July 29, 2014, 3:36 PM
 */

//System Level Libraries

//User Libraries

//Global Constants

//Function Prototypes

//Execution Begins Here:
    //Declare and Initialize Variables

    //Display Menu

    //Welcome Message

    //Initialize board
    //open file
    //Input data from file to board while i is less than 3
    //Close file
```

```
27
28      //Do this while user decides to play again
29          //While game is not over
30             //Display Grid
31             //User takes turn
32             //Switches between players 1 & 2
33
34
35          //If game is over, output message
36              //Display final board that ends game
37              //Displays winner
38              //Keep track of scores
39                  //Add scores for player 1 if player 1 wins
40                  //Add scores for player 2 if player 2 wins
41                  //Don't add any scores for either players if game is a draw
42                  //Output scores to file
43              //Ask if player wants to play again
44              //Sorts grid and prints out dance before board resets
45              //Reset board so user can play again
46
47      //End of main
48
49  //Function to display grid
50      //display board and variables
51
52  //Displays welcome message as well as randomly chooses who plays first
53      //Welcome player and output which player is what mark
54      //Set random seed
55      //Determine who will go first - generate random number from 1 to 2
56          //Set this value to a bool
57
58      //If bool is true
59          //Player 1 gets to play first
60          //It not true, player 2 gets to play first
61
62  //Places mark on board
63      //While number is from 1-9
64      //Get number from user
65      //Convert this number to coordinates on board
66
67      //Player 1
68          //Output an X
69      //Player 2
70          //Output an O
71
72      //Switch between player 1 and 2
73
74  //Determines whether game is over
75      //As long as integer i is less than 3, keep repeating
76          //If there are the same marks adjacent to each other horizontally,
77          //diagonally, or vertically, then the game is over
78          //Game will also end if all spots are taken up
79
80
81  //Determines winner
82      //If there are X's vertically, horizontally, or diagonally adjacent, then
83      //player 1 wins
```

```
84      //If there are X's vertically, horizontally, or diagonally adjacent, then
85      //player 2 wins
86      //If spots are all taken up and there are no marks next to one another,
87      //then no one wins
88
89  //Menu
90      //Show user options
91      //If they choose 1, play game
92      //If they choose 2, display rules then play game
93
94  //Resets board
95      //Set each coordinate equal to a number in increasing order from 1 to 9
96
97
98  //Reads scores to file
99      //Open file
100     //Output scores to file
101     //Compares scores between player 1 and player 2
102         //If player 1's scores is less than player 2's scores, then
103         //tell player 1 to step up their game
104         //If player 2's scores is less than player 1's scores, then
105         //tell player 2 to step up their game
106         //If scores are equal, tell them they are tied
107     //Close file
108
109 //Sorts grid out after game is over
110     //Go through each element of array, compare each value to each other
111     //If value 1 is greater than value 2, then swap values
112     //Display grid after sorting
113     //Stall printing of grid
114
115 //Used to swap values in bubble sort
116     //switch one value with another using a temporary variable
117
118 //Stalls printing of the graph each time it sorts itself
119     //count numbers from 0 to 100000000 times
120
121
122
123 //Converts user's choice to coordinates on grid
124     //if the user chooses 1
125         //set coordinates
126     //if the user chooses 2
127         //set coordinates
128     //if the user chooses 3
129         //set coordinates
130     //if the user chooses 4
131         //set coordinates
132     //if the user chooses 5
133         //set coordinates
134     //if the user chooses 6
135         //set coordinates
136     //if the user chooses 7
137         //set coordinates
138     //if the user chooses 8
139         //set coordinates
140     //if the user chooses 9
```

```
141         //set coordinates
142
```

## CODE

```
  1 /*
  2  * File:   main.cpp
  3  * Author: Anh Vu
  4  * Project 2
  5  * Created on July 25, 2014, 10:17 PM
  6  */
  7
  8 //System Level Libraries
  9 #include <iostream>
 10 #include <ctime>
 11 #include <cstdlib>
 12 #include <fstream>
 13 using namespace std;
 14
 15 //User Libraries
 16
 17 //Global Constants
 18 const int COL=3;        //Size of columns of grid
 19
 20 //Function Prototypes
 21 void welcome();
 22 void displayGrid(char[][COL],const int);   //Displays Tic-Tac-Toe board
 23 void takeTurn(char[][COL],const int,bool); //Places appropriate mark on board
 24 bool gameOver(char[][COL],const int);  //Determines whether or not game is over
 25 short winner(char[][COL],const int);       //Determines winner
 26 void menu();                               //Displays menu
 27 void reset (char[][COL],const int);        //Resets board
 28 void file (int,const int);                 //Outputs scores to file
 29 void sort(char[][COL],const int);          //Sorts board
 30 void swap(char&,char&);                    //Swap values used for sort
 31 void stall();                 //Stalls program so it can print out sorts slower
 32 void conversion(char,int&,int&);        //Converts user's choice to coordinates
 33
 34
 35 //Execution Begins Here:
 36 int main(int argc, char** argv) {
 37     //Declare and Initialize Variables
 38     const int ROW=3;                //Size of rows
 39     char grid[ROW][COL];            //Array used to print grid
 40     bool p1;                        //Used to determine who's turn it is
 41     short win;                      //Used to determining winner
 42     unsigned short score1=0;        //Calculate score for player 1
 43     unsigned short score2=0;        //Calculate score for player 2
 44     char again;                     //Whether players want to play again
 45     bool over;                      //Determines if game is over
 46     bool first;                     //Used to switch between players
 47
 48     //Display Menu
 49     menu();
 50
 51     //Welcome Message
 52     welcome();
 53
```

```
54    //Initialize board
55    //Open and input data from file
56    ifstream board;
57    board.open("board.txt");
58
59    //Read Data into array
60    for (int i=0; i<ROW; i++){
61        for (int j=0; j<COL; j++)
62            board>>grid[i][j];
63     }
64
65     //Close file
66     board.close();
67
68     //Reiterates game until players decide to stop
69     do{
70         //Game continues until it is over
71         do{
72             //Display Grid
73             displayGrid(grid,ROW);
74             //User takes turn
75             takeTurn(grid,ROW,first);
76             //Switches between players 1 & 2
77             if(first){
78                 first=false;
79             }else{
80                 first=true;
81             }
82         }while(!gameOver(grid, ROW));
83
84         //Determines if game is over & outputs game over message
85         gameOver(grid, ROW);
86         if (over=true){
87             cout<<"\n*********************************************\n";
88             cout<<"                  Game Over!          \n";
89             cout<<"*********************************************\n";
90         }
91
92         //Display final board that ends game
93         displayGrid(grid, ROW);
94
95         //Displays winner
96         win = winner(grid,ROW);
97
98         //Keep track of scores
99         //win=1, player 1 wins
100        //win=0, player 2 wins
101        //win=-1; no one wins
102        if (win==1){
103            score1++;
104        }else if (win==0){
105            score2++;
106        }else if (win==-1){
107            score1+=0;
108            score2+=0;
109        }
110
```

```
111         //Output scores to file
112         file (score1,score2);
113 
114         //Ask if player wants to play again
115         cout<<"Play again (Y/N)? ";
116         cin>>again;
117 
118         //Sorts grid and prints out dance before board resets
119         sort(grid,ROW);
120 
121         //Reset board so user can play again
122         reset(grid, ROW);
123     }while((again=='Y')||(again=='y'));
124 
125     //Exit Stage Right!
126     return 0;
127 }
128 
129 //Function displays grid
130 void displayGrid(char grid[][COL], const int ROW){
131     //Board with variables at designated positions
132     cout<<"             |           |           "<<endl;
133     cout<<"       "<<grid[0][0]<<"     |     "<<grid[0][1]<<"     |     ";
134     cout<<grid[0][2]<<endl;
135     cout<<"   __________|___________|__________"<<endl;
136     cout<<"             |           |           "<<endl;
137     cout<<"       "<<grid[1][0]<<"     |     "<<grid[1][1]<<"     |     ";
138     cout<<grid[1][2]<<endl;
139     cout<<"   __________|___________|__________"<<endl;
140     cout<<"             |           |           "<<endl;
141     cout<<"       "<<grid[2][0]<<"     |     "<<grid[2][1]<<"     |     ";
142     cout<<grid[2][2]<<endl;
143     cout<<"             |           |           "<<endl;
144 }
145 
146 //Displays welcome message as well as randomly chooses who plays first
147 void welcome(){
148     //Welcome player and output which player is what mark
149     cout<<"Welcome to Tic-Tac-Toe! First player will be randomly chosen.";
150     cout<<" May the best man win!"<<endl;
151     cout<<"Player 1= X"<<endl;
152     cout<<"Player 2= O"<<endl<<endl;
153 
154     //Set random seed
155     srand(static_cast<unsigned int>(time(0)));
156 
157     //Determine who will go first
158     bool first=rand()%2;
159 
160     //Output who will go first
161     if(first){
162         cout<<"Congratulations Player 1, you get to play first!";
163         cout<<endl<<endl;
164     }else{
165         cout<<"Congratulations Player 2, you get to play first!";
166         cout<<endl<<endl;
167     }
```

```
168 }
169
170 //Places mark on board
171 void takeTurn(char grid[][COL], const int ROW, bool p1){
172     //Declare Variables
173     char ch;            //User's choice
174     int row;            //Row coordinate
175     int column;         //Column coordinate
176
177     //Gather Data Input
178     //Takes user's choices as long as it's valid
179     do{
180         if(p1){
181             cout<<"PLAYER 1'S TURN"<<endl;
182         }else{
183             cout<<"PLAYER 2'S TURN"<<endl;
184         }
185         cout<<"Enter number: ";
186         cin>>ch;
187         //Converts user's choice to coordinate on game board
188         conversion(ch,row,column);
189     }while(row==3||column==3);
190
191     //If player one makes mark, place an X. If player 2, place O.
192         if(p1){
193         grid[row][column] = 'X';
194         }else{
195         grid[row][column] = 'O';
196         }
197 }
198
199 //Determines whether game is over
200 bool gameOver(char grid[][COL], const int ROW){
201     //Declare Variables
202     bool over=false;
203
204     //Game over conditions
205     //Checks for rows
206     for(int i=0; i<ROW; i++){
207         if((grid[i][0]==grid[i][1])&&(grid[i][0]==grid[i][2]))
208             over=true;
209     }
210
211     //Checks for columns
212     for (int j=0; j<COL; j++){
213         if ((grid[0][j]==grid[1][j])&&(grid[0][j]==grid[2][j]))
214             over=true;
215     }
216
217     //Checks for diagonals
218     if ((grid[0][0]==grid[1][1])&&(grid[0][0]==grid[2][2]))
219             over=true;
220     else if ((grid[0][2]==grid[1][1])&&(grid[0][2]==grid[2][0]))
221             over=true;
222
223     //Checks for a draw
224     else if (((grid[0][0]=='X')||(grid[0][0]=='O'))&&
```

```
            ((grid[0][1]=='X')||(grid[0][1]=='O'))&&
            ((grid[0][2]=='X')||(grid[0][2]=='O'))&&
            ((grid[1][0]=='X')||(grid[1][0]=='O'))&&
            ((grid[1][1]=='X')||(grid[1][1]=='O'))&&
            ((grid[1][2]=='X')||(grid[1][2]=='O'))&&
            ((grid[2][0]=='X')||(grid[2][0]=='O'))&&
            ((grid[2][1]=='X')||(grid[2][1]=='O'))&&
            ((grid[2][2]=='X')||(grid[2][2]=='O')))
            over=true;

    //Over=true, Game over
    //Over=false, game not over
    return over;
}
//Determines winner
short winner(char grid[][COL], const int ROW){
    //Will be used to determine who wins
    short win=-1;

    //Loop through rows and columns to see who wins
    for(int i=0; i<ROW; i++){
        if ((grid[i][0]=='X')&&(grid[i][1]=='X')&&(grid[i][2]=='X')){
            cout<<"Player 1 wins!"<<endl;
            win=1;
        }else if ((grid[i][0]=='O')&&(grid[i][1]=='O')&&(grid[i][2]=='O')){
            cout<<"Player 2 wins!"<<endl;
            win=0;
        }else if ((grid[0][i]=='X')&&(grid[1][i]=='X')&&(grid[2][i]=='X')){
            cout<<"Player 1 wins!"<<endl;
            win=1;
        }else if ((grid[1][i]=='O')&&(grid[1][i]=='O')&&(grid[2][i]=='O')){
            cout<<"Player 2 wins!"<<endl;
            win=0;
        }
    }

    //Used to find wins in diagonals. Will only execute if statements above
    //aren't true
    if(!(win==1||win==0)){
        if ((grid[0][0]=='X')&&(grid[1][1]=='X')&&(grid[2][2]=='X')){
                cout<<"Player 1 wins!"<<endl;
                win=1;
        }else if ((grid[0][0]=='O')&&(grid[1][1]=='O')&&(grid[2][2]=='O')){
                cout<<"Player 2 wins!"<<endl;
                win=0;
        }else if ((grid[0][2]=='X')&&(grid[1][1]=='X')&&(grid[2][0]=='X')){
                cout<<"Player 1 wins!"<<endl;
                win=1;
        }else if ((grid[0][2]=='O')&&(grid[1][1]=='O')&&(grid[2][0]=='O')){
                cout<<"Player 2 wins!"<<endl;
                win=0;
        }else{
                cout<<"No one wins."<<endl;
                win=-1;
        }
    }

```

```
282     //win=1, player 1 wins
283     //win=0, player 2 wins
284     //win=-1, draw
285     return win;
286 }
287 
288 //Menu
289 void menu(){
290     //Declare Variables
291     char choice;
292 
293     //Prompt user to input choice
294     do{
295         cout<<"Main Menu:  "<<endl;
296         cout<<"[1] Play Game"<<endl;
297         cout<<"[2] Rules"<<endl;
298         cin>>choice;
299     }while(choice<'1'||choice>'2');
300 
301     //Output options
302     switch(choice){
303         case '1':
304             cout<<endl;
305             break;
306         case '2':
307             cout<<"Rules:  "<<endl;
308             cout<<"1. Decide who will be Player 1 or Player 1"<<endl;
309             cout<<"2. When making your move, choose a number (1-9) to place ";
310             cout<<"your mark. "<<endl;
311             cout<<"4. If you attempt to choose a number lower than 1 or ";
312             cout<<"higher than 9 or if the space is already taken, you will";
313             cout<<" be asked to choose another spot."<<endl;
314             cout<<"5. Try to get all three of your marks in a row, ";
315             cout<<" column, or diagonal row before the other player does.\n";
316             cout<<"6. After each game, choose 'Y' if you want to play again";
317             cout<<" or 'N' if you don't. "<<endl;
318             cout<<"7. If you want to see a tally of your scores, see the ";
319             cout<<"text file entitled scores."<<endl;
320             cout<<"8. Have fun!"<<endl<<endl;
321             break;
322     }
323 }
324 
325 //Resets board
326 void reset (char grid[][COL], const int ROW){
327     int num=1;
328     for (int i=0; i<ROW; i++){
329         for (int j=0; j<COL; j++ ){
330             grid[i][j]=(num+'0');
331             num++;
332         }
333     }
334 }
335 
336 //Reads scores to file
337 void file (int score1, int score2){
338     //Output scores to file
```

```
339     ofstream output;
340     output.open ("scores.txt");
341     output<<"Player 1: "<<score1<<" wins & "
342           <<"Player 2: "<<score2<<" wins...........";
343 
344     //Compares scores between player 1 and player 2
345     if(score1<score2)
346         output<<"Player 1, you need to step up your game!\n";
347     else if (score2<score1)
348         output<<"Player 2, you need to step up your game!\n";
349     else
350         output<<"Player 1 and Player 2, you are tied.\n";
351 
352     //Close File
353     output.close ();
354 
355 }
356 
357 //Sorts grid out after game is over
358 void sort(char grid[][COL], const int ROW){
359     //Declare Variables
360     const int TOTAL = 9;        //Number of elements in the board
361 
362     //Loops through enter array to bubble sort elements
363     for(int i=0; i<TOTAL; i++){
364         for(int j=i+1;j<TOTAL;j++){
365             if(grid[0][i]>grid[0][j]){
366                 //Swap function
367                 swap(grid[0][i],grid[0][j]);
368                 //Displays new grid after each sort
369                 displayGrid(grid,ROW);
370                 //Stalls program in between each sort
371                 stall();
372                 cout<<endl;
373             }
374         }
375     }
376 }
377 
378 //Used to swap values in bubble sort
379 void swap(char& x, char& y){
380     //Declare Variables
381     char temp;          //Temporary Variable
382     temp=x;
383     x=y;
384     y=temp;
385 }
386 
387 //Stalls printing of the graph each time it sorts itself
388 void stall(){
389     for(int i=0;i<100000000;i++);
390 }
391 
392 //Converts user's choice to coordinates on grid
393 void conversion(char ch, int &r,int &c){
394     switch(ch){
395         case '1': r=0;c=0;break;
```

```
396         case '2': r=0;c=1;break;
397         case '3': r=0;c=2;break;
398         case '4': r=1;c=0;break;
399         case '5': r=1;c=1;break;
400         case '6': r=1;c=2;break;
401         case '7': r=2;c=0;break;
402         case '8': r=2;c=1;break;
403         case '9': r=2;c=2;break;
404         default: r=3;c=3;
405     }
406 }
407
408
```

## REFERENCES

[1]"Tic Tac Toe." Http://kidavalanche.wordpress.com/2010/02/03/tic-tac-toe-magic-trick prediction/. N.p., n.d. Web.