Project 2

Tic-Tac-Toe

(With AI)

CSC-5/40375

Name: Fuaad Khan

Date: 02/13/14

**Introduction**

Title: Tic-Tac-Toe

Tic-tac-toe (or Noughts and crosses, Xs and Os) is a two player game that involves X’s and O’s. Each player take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game. If there is now winner, it becomes a tie game, or “Cat’s Game.”

**Summary**

Project Size: About 378 lines

Number of Variables: Around 20

In my first project, I programmed a tic-tac-toe game using a two-dimension array in order for each player to place a piece on the grid, or playing board. After going through the processes of the game in my head I came to realize that this was a very inefficient way of programming the game. When made into a two-dimensional array, the user needs to input coordinates i.e. rows and columns. In my second project I had the game made into a regular array where the user inputs a single number that corresponds with the umber on the game board.

The major difference from my last project and this project is my use if Artificial Intelligence, or AI. These AI’s were used as opponents for a user to play against. The user was given the option of playing between an easy AI and an advanced AI. There are extreme differences between both the AI. The easy AI is basically one line of code. It places a piece in random spaces if they are available. However, the advanced AI reads its moves and the user’s move before placing a piece on the board. This took most of my time programming, and is the focus of this project.

Another major difference to how I set the game up is how the wins would be calculated. In my previous two-dimensional array project, I gave each piece a numbered value. “X’s” were 1 and “O’s” were -1. And whenever the horizontal, vertical, or diagonal sums were equal to 3 or -3, one of the players won. In this project, I made the program check if the characters on the horizontal, vertical, and diagonal axis were equivalent and give the winner as a result.

The user also has the option of playing against a second user. This is similar to my first project.

**Major Variables:**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Variable Name | Description | Location |
| int | gmNum | Number of games played. Increments when the User wants to play again. Helps calculate who is going to go first. | int main(); |
| char | space | This variables displays an ‘X’ or an ‘O’ for the player’s pieces. | int main (); |
| string | p1Name | Name for the first (User) player. | int main(); |

**Major Constructs:**

|  |  |
| --- | --- |
| New Syntax, Key Words | Location |
| Equality operators and relational operators  (==, !=, >, <, >=, <=) | int advAI(char a[]) |
| Arithmetic operators (+, -, \*, /, %) | char plyrTrn(int &p) |
| If/else statements | char gtSpc(char a[],int p, string p1, string p2,int numP) |
| For loops | int main(); |
| Increment operator (++) | int main(); |
| Array | int advAI(char a[]) |
| Two-dimensional array | void rcrdScr(int s[][LOSSES],int,string p1,string p2,int g,int d) |
| Functions | Int main(); |
| Pass by value | void gameBoard(char a[],string p1, string p2,int g) |
| Pass by reference | char plyrTrn(int &p) |
| Returning primitive data types | char plyrTrn(int &p) |
| Reading and Writing into a file | void rcrdScr(int s[][LOSSES],int,string p1,string p2,int g,int d) |

Flow chart provided in project folder.