6/7/2020

Professor Mark Lehr

Project 2

Connect 4 Game Updated

Alcala, Daniel

2020

Introduction

This program is an updated version of the older project, which was the game called Connect 4. This game is a 1 on 1 game where two players put tokens on a board to try and get a diagonal, horizontal, or vertical row of four. The tokens can only be entered at the top, so players can cancel each other’s potential combinations. It involves strategy with a bit trickery.

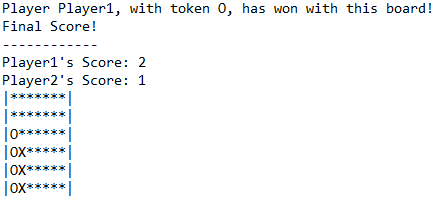
Summary

In the new updated version, the total amount of lines comes to about 600 lines of code and comments. This improvement allowed the program to have several new features, including option of token, win counter, and new way to handle the player objects. Overall, this project was much harder than the original project. The main reason for me not being able to finish the project was due to the fact that I did not give myself enough time to think about how to implement all the new material learned into my code. One such factor was implementing templates into the program. I had several different ideas on how to do so, but after trying each one, lead to the program becoming more convoluted and even broken. Another concept I wish I had more time to deal with was polymorphism. Since I spent so much time on the other concepts, I never really had the chance to try and implement this topic into it.

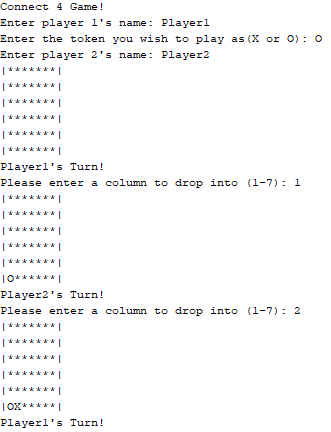
Description

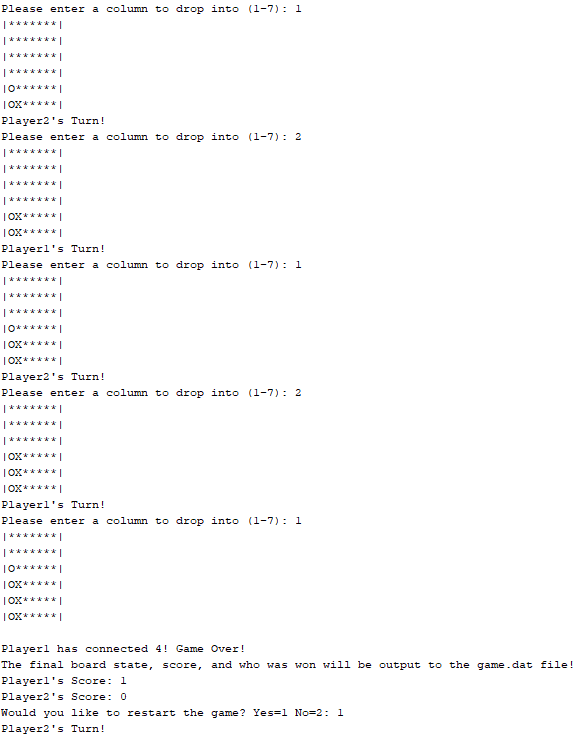
In terms of how I went solving many of the issues, I tried to apply what I learned from the homework and the book and applied it to the code I already had. Some of the concepts I was most familiar with, such as classes, inheritance, and overloading operators, were able to implement in the code easier. I also tried to look at my game and see how I could improve it. Things like a score counter and token choice seemed like good ways to implement the new concepts with the updated version.

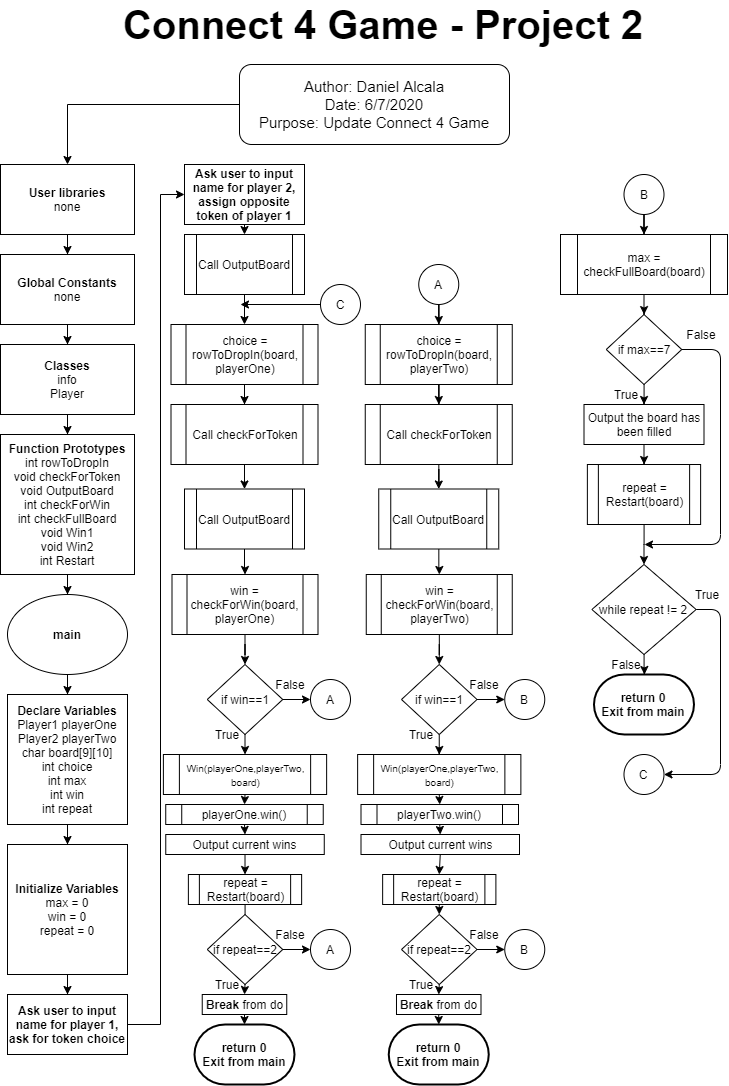
Here is a screenshot of the game.dat file that outputs after a winner has been found:

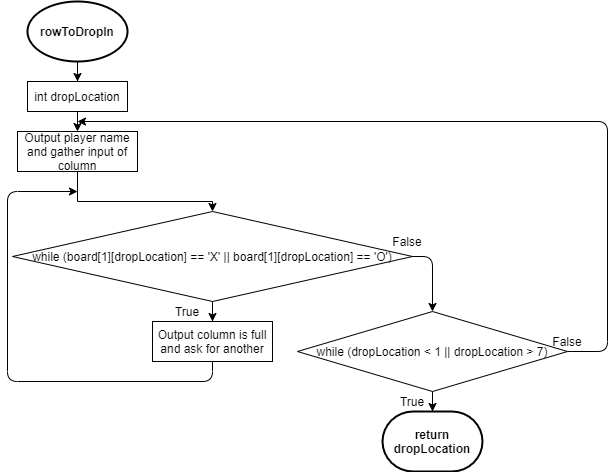


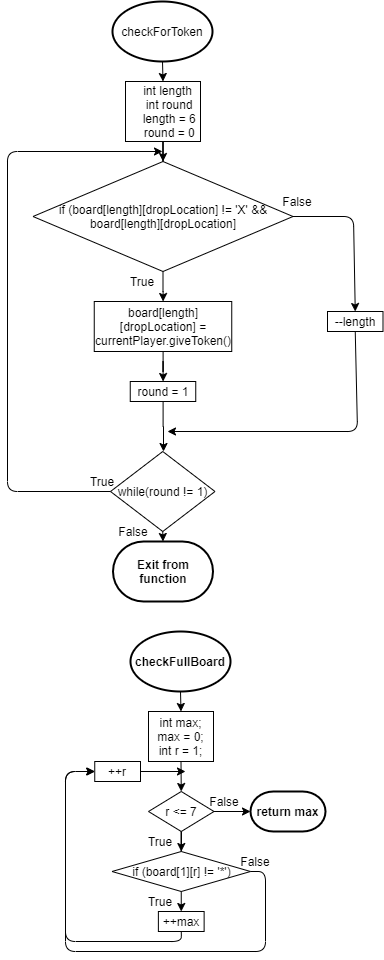
And here is a small snippet of some of the output:

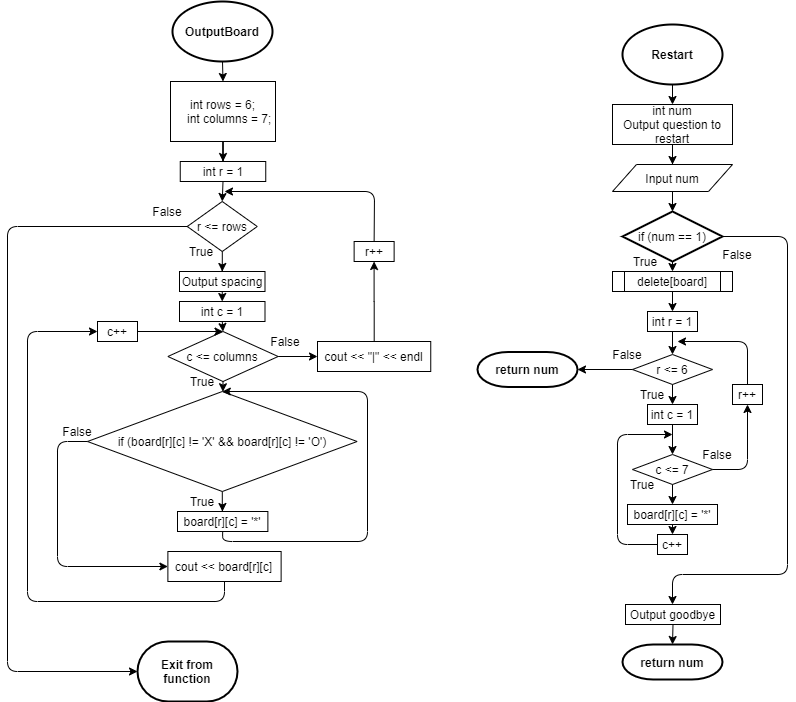


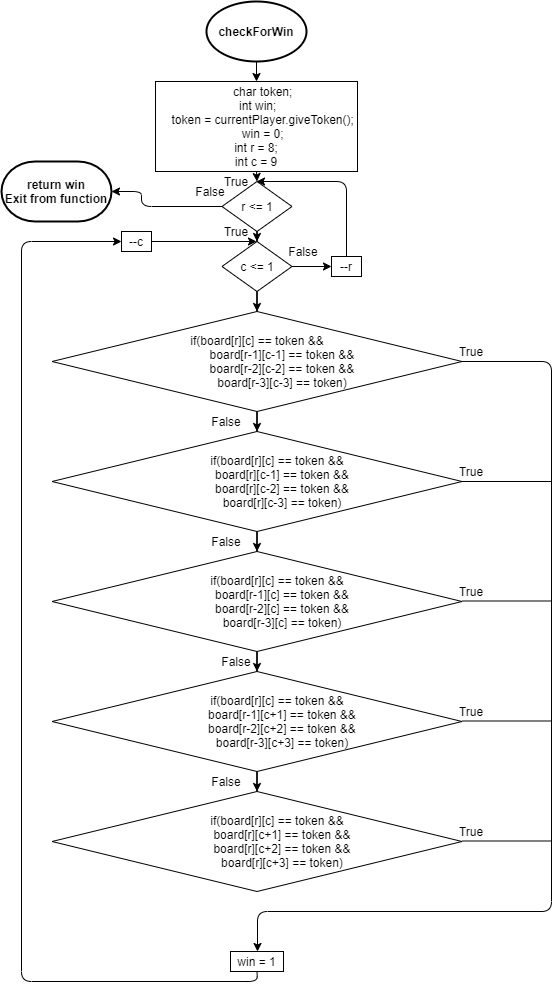


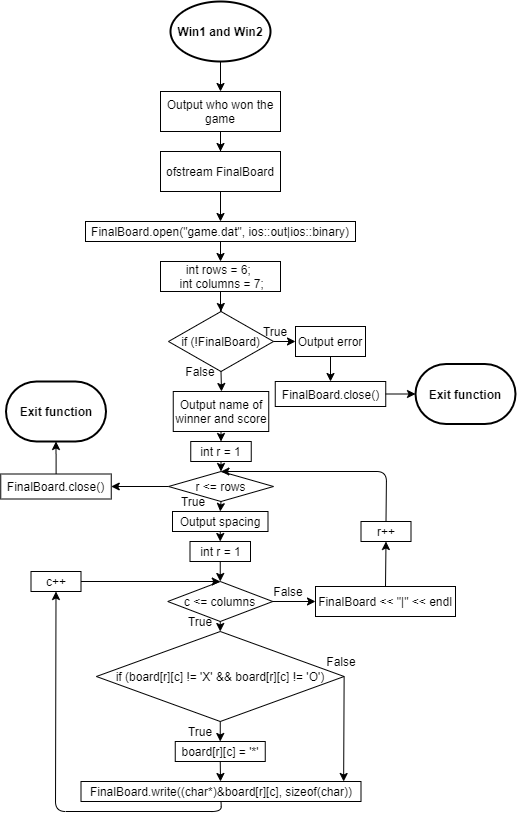
Flowchart

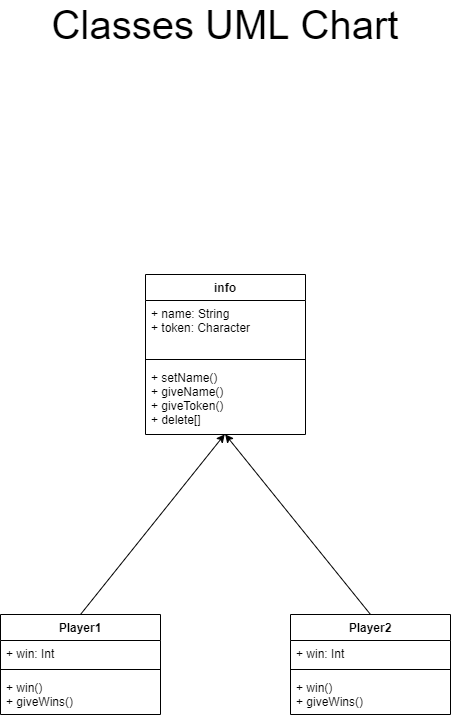












Pseudocode

*Initialize*

*Ask user for name 1*

*Ask name 1 to choose token*

*Ask user for name 2*

*Assign name 2 to opposite token*

*Output board*

*Do*

*Let player 1 pick row*

*Place token for player 1*

*Output board*

*Check for player 1 win and store it*

*If player 1 equals a win*

*Output player 1 has won and write to .dat file*

*Ask the user if they wish to repeat the game*

*If user wants to repeat*

*Delete board and start game again, else end game*

*Let player 2 pick row*

*Place token for player 2*

*Output board*

*Check for player 2 win and store it*

*If player 2 equals a win*

*Output player 2 has won and write to .dat file*

*Ask the user if they wish to repeat the game*

*If user wants to repeat*

*Delete board and start game again, else end game*

*Find how many rows have maxed out with tokens*

*If tokens have maxed out each row*

*Output the board has been filled*

*Ask the user if they wish to repeat the game*

*While users wish to repeat the game*

*Terminate program*

Program

/\*

\* File: main.cpp

\* Author: Daniel Alcala

\* Date: June 7, 2020

\* Purpose: Connect 4 Game Project 2 V6

\*/

#ifndef HEADER\_H

#define HEADER\_H

#include <iostream> //I/O Library

#include <iomanip>

#include <string> //Strings

#include <fstream> //File Output/Input

using namespace std; //Libraries compiled under std

class info

{

private:

string name;

char token;

public:

void setName(string na)

{

name = na;

}

void setToken(char to)

{

token = to;

}

string giveName()const

{

return name;

}

char giveToken()const

{

return token;

}

void operator delete[](void\* ptr)

{

cout << "Now deleting the board and refilling!";

free(ptr);

}

};

class Player : public info

{

private:

int wins = 0;

public:

void win()

{

wins++;

}

int giveWins() const

{

return wins;

}

};

int rowToDropIn (char board[][10], info currentPlayer);

void checkForToken (char board[][10], info currentPlayer, int dropLocation);

void OutputBoard (char board[][10]);

int checkForWin (char board[][10], info currentPlayer);

int checkFullBoard (char board[][10]);

void Win1 (Player p1, Player p2, char board [][10]);

void Win2 (Player p1, Player p2, char board [][10]);

int Restart (char board[][10]);

#endif /\* HEADER\_H \*/

/\*

\* File: main.cpp

\* Author: Daniel Alcala

\* Date: June 7, 2020

\* Purpose: Connect 4 Game Project 2 V6

\* Version 6 Info: This is the final version of the game.

\*/

//System Level Libraries

#include "header.h"

//Execution Begins Here!

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

//Random Number Seed Set Here

//Variable Declarations

Player playerOne,playerTwo;

char board[9][10];

string name;

char token;

int choice,

max,

repeat,

win;

//Variable Initialization

max = 0;

repeat = 0;

win = 0;

//Output To Gather Information

cout << "Connect 4 Game!" << endl;

cout << "Enter player 1's name: ";

getline(cin, name);

playerOne.setName(name);

cout << "Enter the token you wish to play as(X or O): ";

cin >> token;

cin.ignore();

token = toupper(token);

while (token != 'X' && token != 'O')

{

cout << "Incorrect token choice! Try again: ";

cin >> token;

cin.ignore();

}

playerOne.setToken(token);

cout << "Enter player 2's name: ";

getline(cin, name);

playerTwo.setName(name);

if (token == 'X')

{

token = 'O';

playerTwo.setToken(token);

}

else if (token == 'O')

{

token = 'X';

playerTwo.setToken(token);

}

OutputBoard(board);

do

{

choice = rowToDropIn(board, playerOne);

checkForToken(board, playerOne, choice);

//Display Outputs

OutputBoard(board);

win = checkForWin(board, playerOne);

if (win == 1)

{

playerOne.win();

Win1(playerOne, playerTwo, board);

cout << playerOne.giveName() << "'s Score: " << playerOne.giveWins() << endl;

cout << playerTwo.giveName() << "'s Score: " << playerTwo.giveWins() << endl;

repeat = Restart(board);

if (repeat == 2)

{

break;

}

}

choice = rowToDropIn(board, playerTwo);

checkForToken(board, playerTwo, choice);

//Display Outputs

OutputBoard(board);

win = checkForWin(board, playerTwo);

if (win == 1)

{

playerTwo.win();

Win2(playerOne, playerTwo, board);

cout << playerOne.giveName() << "'s Score: " << playerOne.giveWins() << endl;

cout << playerTwo.giveName() << "'s Score: " << playerTwo.giveWins() << endl;

//Clean Up

repeat = Restart(board);

if (repeat == 2)

{

break;

}

}

max = checkFullBoard(board);

if (max == 7)

{

cout << "The board has been completely filled up! The game is a draw!" << endl;

repeat = Restart(board);

}

}while (repeat != 2);

//Exit stage right!

return 0;

}

//This function outputs the entire board, with the sides. Empty spaces

//are seen as \*, while the tokens are X and O for each player.

#include "header.h"

void OutputBoard (char board[][10])

{

int rows = 6;

int columns = 7;

for (int r = 1; r <= rows; r++)

{

cout << "|";

for (int c = 1; c <= columns; c++)

{

if (board[r][c] != 'X' && board[r][c] != 'O')

{

board[r][c] = '\*';

}

cout << board[r][c];

}

cout << "|" << endl;

}

}

//This function asks the user if they would like to restart the game.

//If they do, then the board will reset back to the original game state.

//If not, the function will terminate the game.

#include "header.h"

int Restart (char board[][10])

{

int num;

cout << "Would you like to restart the game? Yes=1 No=2: ";

cin >> num;

if (num == 1)

{

delete []board;

for(int r =1; r <= 6; r++)

{

for (int c = 1; c <= 7; c++)

{

board[r][c] = '\*';

}

}

}

else

{

cout << "Goodbye!" << endl;

}

return num;

}

//This function mainly outputs the board to a .dat file and who won that game

//board.

#include "header.h"

void Win1 (Player p1, Player p2, char board[][10])

{

cout << endl << p1.giveName() << " has connected 4! Game Over!" << endl;

ofstream FinalBoard;

FinalBoard.open("game.dat", ios::out|ios::binary);

int rows = 6;

int columns = 7;

if (!FinalBoard)

{

cout << "Error opening file!" << endl;

}

else

{

FinalBoard << "Player " << p1.giveName() << ", with token " << p1.giveToken() << ", has won with this board!" << endl;

FinalBoard << "Final Score!" << endl;

FinalBoard << "------------" << endl;

FinalBoard << p1.giveName() << "'s Score: " << p1.giveWins() << endl;

FinalBoard << p2.giveName() << "'s Score: " << p2.giveWins() << endl;

for (int r = 1; r <= rows; r++)

{

FinalBoard << "|";

for (int c = 1; c <= columns; c++)

{

if (board[r][c] != 'X' && board[r][c] != 'O')

{

board[r][c] = '\*';

}

FinalBoard.write((char\*)&board[r][c], sizeof(char));

}

FinalBoard << "|" << endl;

}

}

FinalBoard.close();

cout << "The final board state, score, and who was won will be output to the game.dat file!" << endl;

}

//This function checks for any tokens that might be in the columns the

//user wishes to input into and would place a token in that column if

// avaliable.

#include "header.h"

void checkForToken (char board[][10], info currentPlayer, int dropLocation)

{

int length;

int round;

length = 6;

round = 0;

do

{

if (board[length][dropLocation] != 'X' && board[length][dropLocation] != 'O')

{

board[length][dropLocation] = currentPlayer.giveToken();

round = 1;

}

else

{

--length;

}

}while(round != 1);

}

//This function checks all possible combinations for the player to win

//at the game. It will return a "true" if a combination has been found.

#include "header.h"

int checkForWin (char board[][10], info currentPlayer)

{

char token;

int win;

token = currentPlayer.giveToken();

win = 0;

for(int r = 8; r >= 1; --r)

{

for(int c = 9; c >= 1; --c)

{

if(board[r][c] == token &&

board[r-1][c-1] == token &&

board[r-2][c-2] == token &&

board[r-3][c-3] == token)

{

win = 1;

}

if(board[r][c] == token &&

board[r][c-1] == token &&

board[r][c-2] == token &&

board[r][c-3] == token)

{

win = 1;

}

if(board[r][c] == token &&

board[r-1][c] == token &&

board[r-2][c] == token &&

board[r-3][c] == token)

{

win = 1;

}

if(board[r][c] == token &&

board[r-1][c+1] == token &&

board[r-2][c+2] == token &&

board[r-3][c+3] == token)

{

win = 1;

}

if(board[r][c] == token &&

board[r][c+1] == token &&

board[r][c+2] == token &&

board[r][c+3] == token)

{

win = 1;

}

}

}

return win;

}

//This function checks to see if all rows are full or not.

#include "header.h"

int checkFullBoard (char board[][10])

{

int max;

max = 0;

for (int r = 1; r <= 7; ++r)

{

if (board[1][r] != '\*')

{

++max;

}

}

return max;

}

//This asks the user to input which row they wish to drop their token into.

//If the row is full then it will ask the user for a different row.

#include "header.h"

int rowToDropIn (char board[][10], info currentPlayer)

{

int dropLocation;

do

{

cout << currentPlayer.giveName() << "'s Turn!" << endl;

cout << "Please enter a column to drop into (1-7): ";

cin >> dropLocation;

while (board[1][dropLocation] == 'X' || board[1][dropLocation] == 'O')

{

cout << "That column is full, please try again: ";

cin >> dropLocation;

}

}while (dropLocation < 1 || dropLocation > 7);

return dropLocation;

}

//This function mainly outputs the board to a .dat file and who won that game

//board.

#include "header.h"

void Win2 (Player p1, Player p2, char board[][10])

{

cout << endl << p2.giveName() << " has connected 4! Game Over!" << endl;

ofstream FinalBoard;

FinalBoard.open("game.dat", ios::out|ios::binary);

int rows = 6;

int columns = 7;

if (!FinalBoard)

{

cout << "Error opening file!" << endl;

}

else

{

FinalBoard << "Player " << p2.giveName() << ", with token " << p2.giveToken() << ", has won with this board!" << endl;

FinalBoard << "Final Score!" << endl;

FinalBoard << "------------" << endl;

FinalBoard << p1.giveName() << "'s Score: " << p1.giveWins() << endl;

FinalBoard << p2.giveName() << "'s Score: " << p2.giveWins() << endl;

for (int r = 1; r <= rows; r++)

{

FinalBoard << "|";

for (int c = 1; c <= columns; c++)

{

if (board[r][c] != 'X' && board[r][c] != 'O')

{

board[r][c] = '\*';

}

FinalBoard.write((char\*)&board[r][c], sizeof(char));

}

FinalBoard << "|" << endl;

}

}

FinalBoard.close();

cout << "The final board state, score, and who was won will be output to the game.dat file!" << endl;

}