Final Project

Connect 4

By Dillin Alexander

Class:

CIS 17A

Course Code:

47975

December 09, 2013

Table of Content

Page 3 Introduction/Description

Page 4 Variables

Page 5 Sample Output

Page 14 Source Code

Page 54 Flowchart

Introduction/Description

Connect four is a classical board game that involves two players who play each other dropping colored game pieces down the column of a board. The person who connects four in a row is the winner, whether it be horizontally, diagonally, or vertically. My virtual connect four does the same thing or at least shares the same concept, letting players choose their own colors and names while the program runs. The program even saves all the data for later viewing in a file in the same folder as the program called “game.txt”. This program is 1,364 lines long and utilizes classes, class objects, file output and inputs, inheritance, and even a two dimensional arrays of objects. The two dimensional array of objects is what is used to create the board for this game. It holds forty two different characters, well they are the same when initialized then are changed when the player chooses a column to place their game piece. When a person choose a column out of the choices their game piece is placed at the lowest possible row as long as the row and column isn’t occupied by other game pieces.

This program wasn’t that difficult to build using the techniques learned in class, the only difficult parts were testing the six by seven board for all the possible winning combinations. This process by itself took about a thousand lines of code and was very time consuming, when loops couldn’t be implemented. Overall I enjoyed the project and learned a lot from completing it and will continue to find ways to make the code work a little better and run a little faster.

Variables

|  |  |  |
| --- | --- | --- |
| Variable, objects, members, structures, classes | | |
| Classes | Color | char clr[2] |
| Winner | int Winner |
| Player | string plyr[2] |
| Structures | Board | char board[6][7] |
| int | num, count, turn, Dcount, Dcount2, Dcount3, Dcount4, Vcount, Hcount, Pcount, Ncount | |
| char | choice, coin, y | |
| string | Stuff, name1, name2, | |
| fstream | outfile, output | Files:  Games.txt,  File.txt,  Player 1.txt,  Player 2.txt. |
| Class Object | color | char clr |
| winner | int win |
| Players | string plyr |
| GameBoard | Board |

Sample Output

Welcome to connect four.

In this game there will be two players who drop

their certain colored game piece down a column

of the gameboard and try to connect four pieces

in a row. The player can connect their pieces

diagonally, horizontally, or vertically. The

first player to four wins.

Lets play!!!!

Enter the name of player 1: Dillin

Enter the name of player 2: Alexander

We will flip a coin to see who goes first!

Player 1 type H for heads or T for tails: H

The coin flip was heads.

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 2

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | O | O | O | O | O | O | O |

6 | O | G | O | O | O | O | O |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 3

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | O | O | O | O | O | O | O |

6 | O | G | P | O | O | O | O |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 4

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | O | O | O | O | O | O | O |

6 | O | G | P | G | O | O | O |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 3

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 2 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | O | O | P | O | O | O | O |

6 | O | G | P | G | O | O | O |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 1

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | O | O | P | O | O | O | O |

6 | G | G | P | G | O | O | O |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 1

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | P | O | P | O | O | O | O |

6 | G | G | P | G | O | O | O |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 2

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 2 |

Diagonal count (Positive slope) = 2 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | O | O | O | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | O | O | O |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 1

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 2 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | O | O | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | O | O | O |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 2

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 3 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | O | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | O | O | O |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 3

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 3 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | O | O | O |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 7

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | O | O | G |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 6

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | O | P | G |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 5

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 2 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | O | O | O | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 4

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 2 |

Vertical count = 1 |

Diagonal count (Positive slope) = 2 |

Diagonal count (Negative slope) = 2 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | P | O | O | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 6

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 2 |

Diagonal count (Negative slope) = 2 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | O | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | P | O | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 2

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 2 |

Diagonal count (Negative slope) = 3 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | P | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | P | O | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 5

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 2 |

Vertical count = 2 |

Diagonal count (Positive slope) = 2 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | P | O | O | O | O | O |

4 | P | G | P | O | O | O | O |

5 | P | G | P | P | G | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 4

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 2 |

Vertical count = 2 |

Diagonal count (Positive slope) = 2 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | P | O | O | O | O | O |

4 | P | G | P | P | O | O | O |

5 | P | G | P | P | G | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 5

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 3 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 3 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | P | O | O | O | O | O |

4 | P | G | P | P | G | O | O |

5 | P | G | P | P | G | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 6

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 1 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | O | P | O | O | O | O | O |

4 | P | G | P | P | G | P | O |

5 | P | G | P | P | G | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Dillin its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 1

G

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 1 |

Vertical count = 1 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 2 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | G | P | O | O | O | O | O |

4 | P | G | P | P | G | P | O |

5 | P | G | P | P | G | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

Alexander its your turn!

What column would you like to drop your piece? (enter a column number)

Column: 3

P

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

horizontal count = 2 |

Vertical count = 4 |

Diagonal count (Positive slope) = 1 |

Diagonal count (Negative slope) = 2 |

-------------------------------------

C: 1 2 3 4 5 6 7

1 | O | O | O | O | O | O | O |

2 | O | O | O | O | O | O | O |

3 | G | P | P | O | O | O | O |

4 | P | G | P | P | G | P | O |

5 | P | G | P | P | G | G | O |

6 | G | G | P | G | G | P | G |

-----------------------------

The winner is Alexander

Congratulations!!!!

Source Code

/\*

\* File: Final Project

\* Author: Dillin Alexander

\* Date: December 9th, 2013

\*/

#include <cstdlib>

#include <iostream>

#include <ctime>

#include <fstream>

#include "GameBoard.h"

#include "Players.h"

#include "Color.h"

#include "Winner.h"

using namespace std;

void displayRules(fstream&,fstream&);

void initBoard(GameBoard[][7]);

void getNames(Players&, fstream&);

bool chooseMove(GameBoard[][7], int,color&,fstream&);

void PlayGame(GameBoard[][7], int, Winner&,color&,fstream&);

int HeadsTails(fstream&);

bool TestScore(GameBoard[][7],int, int,fstream&);

template <class T>

T InputOutput(T infile){

string stuff;

while(getline(infile,stuff)){

cout << stuff << endl;

}

}

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

const int ROW=6,COL=7;

//Declare two dimensional array of objects

GameBoard bd[ROW][COL];

//Declare object with 2 players.

Players names;

//Declare a p winner object

Winner plyr;

//declare a color object

color clr;

//Define integer to hold players turn

int turn;

//declare a file stream object

fstream infile("file.txt",ios::in);

//declare a file stream object

fstream outfile("Game.txt", ios::out);

//display the rules of the game

displayRules(infile,outfile);

//Initialize the game board

initBoard(bd);

//Get the names of the players

getNames(names,outfile);

//Have the players choose a color

clr.setColor();

//randomize who goes first

turn = HeadsTails(outfile);

//play game

PlayGame(bd, turn, plyr, clr, outfile);

infile.close();

outfile.close();

return 0;

}

void initBoard(GameBoard bd[][7]){

//Initialize board pieces

for(int i=0; i < 6; i++){

for(int j=0; j<7; j++){

bd[i][j].initBoard();

}

}

}

void displayRules(fstream &infile, fstream &ouput){

string stuff;

while(getline(infile,stuff)){

cout << stuff << endl;

ouput << stuff << endl;

}

cout << endl;

}

void getNames(Players &names, fstream &output){

string name1,name2;

//Get the players names

cout << "Enter the name of player 1: ";

output << "Enter the name of player 1: ";

getline(cin,name1);

output << name1 << endl;

cout << "Enter the name of player 2: ";

output << "Enter the name of player 2: ";

getline(cin,name2);

output << name2 << endl;

names.setPlayr(name1,name2);

cout << endl;

}

void displayBoard(GameBoard Board[][7], fstream &output){

cout << "C: ";

output << "C: ";

for(int i = 0; i < 7; i++){

cout << i+1 <<" ";

output << i+1 <<" ";

}

cout << endl;

output << endl;

for(int r = 0; r < 6; r++){

for(int c = 0; c < 7; c++){

if(c==0){

cout << r+1;

output << r+1;

}

cout << " | ";

output << " | ";

cout << Board[r][c].compareBoard();

output << Board[r][c].compareBoard();

}

cout<< " |";

output<< " |";

cout << endl;

output<<endl;

}

cout << " -----------------------------" << endl;

output <<" -----------------------------" << endl;

}

int HeadsTails(fstream &output){

//declare two character values

char coin, y;

int x;

//loop until player 1 enters correct choice

do{

cout << "\nWe will flip a coin to see who goes first!" << endl;

output << "\nWe will flip a coin to see who goes first!" << endl;

cout << "Player 1 type H for heads or T for tails: ";

output << "Player 1 type H for heads or T for tails: ";

cin >> coin;

output << coin << endl;

}while(coin != 'H' && coin != 'T');

srand(time(0));

//get random number for coin flip

x=(rand()%2)+1;

//Output what the coin flip was!

cout << "\nThe coin flip was ";

output << "\nThe coin flip was ";

if (x==1){

cout << "heads." << endl;

output << "heads." << endl;

y = 'H';

}

else if (x==2){

cout << "tails." << endl;

output << "tails." << endl;

y = 'T';

}

//set index equal to one

if(coin == y){

x = 1;

}//or set index equal to two

else x = 2;

//return the index of the player who won the coin toss.

return x-1;

}

void PlayGame(GameBoard brd[][7], int x, Winner &plyr, color &clr, fstream &output){

bool flag=false; //set flag to test for winner

//Whoever wins coin toss goes first

chooseMove(brd, x,clr,output);

//output first move

displayBoard(brd, output);

//increment to next player

x++;

//loop till we gone through all pieces on the board.

for(int i = 0; i < 41; i++){

//return to the previous player.

if(x>1)x=0;

//return true if a player wins, else return false.

flag = chooseMove(brd, x, clr,output);

//Display four square board

displayBoard(brd,output);

//If a winner is decided break

if(flag==true){

plyr.AnnceWinner(x);

break;

}

//Increment player count

x++;

//test to break out of loop if winner declared

if (flag==true) i = 45;

}

//If no winner declared then there is a tie.

if(flag==false){

cout << "There is Draw!" << endl;

output << "There is a Draw!" << endl;

}

}

bool chooseMove(GameBoard Board[][7], int x, color &clr, fstream &output){

int col=0, r = 6;

bool doOvr, win = false;

cout << clr.getPlayer(x) << " its your turn!" << endl;

output << clr.getPlayer(x) << " its your turn!" << endl;

do{

doOvr=true;

//Test to validate that the input is between one and seven.

do{

cout << "What column would you like to drop your piece? (enter a column number)" << endl;

output << "What column would you like to drop your piece? (enter a column number)" << endl;

cout << "Column: ";

output << "Column: ";

cin >> col;

output << col << endl;

}while(col!=1&&col!=2&&col!=3&&col!=4&&col!=5&&col!=6&&col!=7);

cout << clr.getColor(x) << endl;

output << clr.getColor(x) << endl;

for(int i = 5; i >= 0; i--){

//test to see if the column is taken

if(Board[i][col-1].compareBoard()== 'O'){

char y = clr.getColor(x);

Board[i][col-1].setBoard(y);

//test to see if the players choice makes four in a row

win = TestScore(Board, i, col-1,output);

i = -1;

}

else if(i == 0){//Test to see if the column is full

cout << "Column full! Enter a different column!";

output << "Column full! Enter a different column!";

doOvr=false;

}

}

r--;//decrement the row.

}while(doOvr == false);

return win;

}

bool TestScore(GameBoard Board[][7], int r, int c, fstream &output){

bool flag = false;

int count = 1;

int Dcount = 1,Dcount2 =1,Dcount3 =1,Dcount4 =1,Vcount = 1;

int Pcount=0,Ncount=0;

int t=r;

//Test to see if the choice is in the first column.

if(c == 0){

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

if(Board[r][c].compareBoard()==Board[i][c].compareBoard()){

count++;

}

else break;

}

}

//Test to see if the choice is in the second column.

if(c == 1){

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

if(Board[r][c].compareBoard()==Board[r][i].compareBoard()){

count++;

}

else break;

}

for(int j = c-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[r][j].compareBoard()){

count++;

}

else break;

}

}

//Test to see if the choice is in the third column.

if(c == 2){

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

if(Board[r][c].compareBoard()==Board[r][i].compareBoard()){

count++;

}

else break;

}

for(int j = c-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[r][j].compareBoard()){

count++;

}

else break;

}

}

//Test to see if the choice is in the fourth column.

if(c == 3){

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

if(Board[r][c].compareBoard()==Board[r][i].compareBoard()){

count++;

}

else break;

}

for(int j = c-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[r][j].compareBoard()){

count++;

}

else break;

}

}

//Test to see if the choice is in the five column.

if(c == 4){

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

if(Board[r][c].compareBoard()==Board[r][i].compareBoard()){

count++;

}

else break;

}

for(int j = c-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[r][j].compareBoard()){

count++;

}

else break;

}

}

//Test to see if the choice is in the sixth column.

if(c == 5){

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

if(Board[r][c].compareBoard()==Board[r][i].compareBoard()){

count++;

}

else break;

}

for(int j = c-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[r][j].compareBoard()){

count++;

}

else break;

}

}

//Test to see if the choice is in the seventh column.

if(c == 6){

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

if(Board[r][c].compareBoard()==Board[r][i].compareBoard()){

count++;

}

else break;

}

for(int j = c-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[r][j].compareBoard()){

count++;

}

else break;

}

}

//Start testing for horizontal connections

if(c>=0&&c<=6){

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

if(Board[r][c].compareBoard()==Board[i][c].compareBoard()){

Vcount++;

}

else break;

}

for(int j = r-1; j>=0; j--){

if(Board[r][c].compareBoard()==Board[j][c].compareBoard()){

Vcount++;

}

else break;

}

}

//Start testing for diagonal connections

//Begin with the first column

if(c == 0 && r==5){

for(int i = c+1; i<6; i++){//Diagonal count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}

}

if(c == 0 && r==4){

for(int i = c+1; i<5; i++){//Diagonal count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}

for(int i = c+1; i<2; i++){//Diagonal count down

t=r;

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

if(c == 0 && r==3){

for(int i = c+1; i<4; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<3; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

if(c == 0 && r==2){

for(int i = c+1; i<3; i++){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<4; i++){

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

if(c == 0 && r==1){

for(int i = c+1; i<2; i++){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<5; i++){

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

if(c == 0 && r==0){

for(int i = c+1; i<6; i++){

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

//Continue diagonal search

//Begin count for second column

if(c == 1 && r==5){

for(int i = c+1; i<6; i++){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}

}

if(c == 1 && r==4){

for(int i = c+1; i<5; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<3; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 1 && r==3){

for(int i = c+1; i<4; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<4; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 1 && r==2){

for(int i = c+1; i<3; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<5; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 1 && r==1){

for(int i = c+1; i<2; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<5; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 1 && r==0){

for(int i = c+1; i<6; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

//Continue diagonal search

//Begin count for third column

if(c == 2 && r==5){

for(int i = c+1; i<6; i++){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}

}

if(c == 2 && r==4){

for(int i = c+1; i<5; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<4; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 2 && r==3){

for(int i = c+1; i<4; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<5; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 2 && r==2){

for(int i = c+1; i<3; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<6; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 2 && r==1){

for(int i = c+1; i<2; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 2 && r==0){

for(int i = c+1; i<6; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

//Continue diagonal search

//Begin count for fourth column

if(c == 3 && r==5){

for(int i = c+1; i<6; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}

}

if(c == 3 && r==4){

for(int i = c+1; i<5; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<5; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

} t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 3 && r==3){

for(int i = c+1; i<4; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<6; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 3 && r==2){

for(int i = c+1; i<3; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 3 && r==1){

for(int i = c+1; i<2; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//cound down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 3 && r==0){

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount4++;

}

else break;

}

}

//Continue diagonal search

//Begin count for fifth column

if(c == 4 && r==5){

for(int i = c+1; i<6; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}

t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

}

}

if(c == 4 && r==4){

for(int i = c+1; i<5; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<6; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 4 && r==3){

for(int i = c+1; i<4; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 4 && r==2){

for(int i = c+1; i<3; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 4 && r==1){

for(int i = c+1; i<2; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

} t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 4 && r==0){

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

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

}

//Continue diagonal search

//Begin count for sixth column

if(c == 5 && r==5){

for(int i = c+1; i<7; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}

}

if(c == 5 && r==4){

for(int i = c+1; i<5; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 5 && r==3){

for(int i = c+1; i<4; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 5 && r==2){

for(int i = c+1; i<3; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 5 && r==1){

for(int i = c+1; i<2; i++){//count up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount++;

}

else break;

}t=r;

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}t=r;

for(int i = c-1; i>=0; i--){//count back up

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 5 && r==0){

for(int i = c+1; i<7; i++){//count down

t++;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount2++;

}

else break;

}

for(int i = r+1; i<6; i++){//count back down

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

//Continue diagonal search

//Begin count for Seventh column

if(c == 6 && r==5){

for(int i = c-1; i<0; i--){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}

}

if(c == 6 && r==4){

for(int i = c-1; i<0; i--){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

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

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 6 && r==3){

for(int i = c-1; i<0; i--){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

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

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 6 && r==2){

for(int i = c-1; i<0; i--){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

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

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 6 && r==1){

for(int i = c-1; i<0; i--){

t--;

if(Board[r][c].compareBoard()==Board[t][i].compareBoard()){

Dcount3++;

}

else break;

}t=c;

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

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

if(c == 6 && r==0){

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

t--;

if(Board[r][c].compareBoard()==Board[i][t].compareBoard()){

Dcount4++;

}

else break;

}

}

//Adding up the slopes

if(Dcount==1)Dcount=0;

if(Dcount4==1)Dcount4=0;

Pcount=Dcount+Dcount4;//Adding up the positive diagonal slope

if(Pcount==0)Pcount=1;

if(Dcount2==1)Dcount2=0;

if(Dcount3==1)Dcount3=0;

Ncount=Dcount2+Dcount3;//Adding up the negative diagonal slope

if(Ncount==0)Ncount=1;

//Output the amount the player got in a row.

cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

output << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "horizontal count = " << count << " |" << endl;

output << "horizontal count = " << count << " |" << endl;

cout << "Vertical count = " << Vcount << " |" << endl;

output << "Vertical count = " << Vcount << " |" << endl;

cout << "Diagonal count (Positive slope) = " << Pcount << " |" << endl;

output << "Diagonal count (Positive slope) = " << Pcount << " |" << endl;

cout << "Diagonal count (Negative slope) = " << Ncount << " |" << endl;

output << "Diagonal count (Negative slope) = " << Ncount << " |" << endl;

cout << "-------------------------------------" << endl<<endl;

output << "-------------------------------------" << endl<<endl;

//if a player gets to four send a flag

if(count >= 4)flag =true;

if(Pcount >= 4)flag =true;

if(Ncount >= 4)flag =true;

if(Vcount >= 4)flag =true;

return flag;

}

Player header and source

#ifndef PLAYERS\_H

#define PLAYERS\_H

#include <string>

class Players{

private:

std::string plyr[2];

public:

Players();

void setPlayr(std::string,std::string);

std::string getPlayer(int);

};

#endif /\* PLAYERS\_H \*/

#include <cstdlib>

#include <iostream>

#include <string>

#include <fstream>

#include "Players.h"

#include "Color.h"

#include "Winner.h"

using namespace std;

Players::Players(){

\*(plyr+0) = "Player 1";

\*(plyr+1) = "Player 2";

}

void Players::setPlayr(string x, string y){

fstream name1, name2;

name1.open("Player 1.txt",ios::out);

name2.open("Player 2.txt",ios::out) ;

name1 << x;

name2 << y;

}

std::string Players::getPlayer(int x){

string file;

fstream name1, name2;

name1.open("Player 1.txt",ios::in);

name2.open("Player 2.txt",ios::in) ;

if(x==0)name1>>file;

if(x==1)name2>>file;

name1.close();

name2.close();

return file;

}

Color header and source

#ifndef COLOR\_H

#define COLOR\_H

#include "Players.h"

class color: public Players{

private:

char clr[2];

public:

color();

void setColor();

char getColor(int);

};

#endif /\* COLOR\_H \*/

#include <cstdlib>

#include <iostream>

#include <string>

#include "Players.h"

#include "Color.h"

#include "Winner.h"

using namespace std;

color::color(){

clr[0] = 'Y';

clr[1] = 'B';

}

void color::setColor(){

int choice;

for(int i = 0; i<2; i++){

do{

cout << getPlayer(i) << " choose a color: " << endl;

cout << "1. Blue" << endl;

cout << "2. Green" << endl;

cout << "3. Yellow" << endl;

cout << "4. Purple" << endl;

cout << "5. White" << endl;

cout <<"Color choice: ";

cin >> choice;

}while(choice<1||choice>5);

switch (choice){

case 1: clr[i] = 'B';

break;

case 2: clr[i]= 'G';

break;

case 3: clr[i] = 'Y';

break;

case 4: clr[i] = 'P';

break;

case 5: clr[i] = 'W';

break;

default: break;

}

cout << endl;

}

}

char color::getColor(int x){

return clr[x];

}

Winner header and source

#ifndef WINNER\_H

#define WINNER\_H

#include "Players.h"

#include <string>

class Winner: public Players{

private:

int winner;

public:

void AnnceWinner(int);

};

#endif /\* WINNER\_H \*/

#include <cstdlib>

#include <iostream>

#include <string>

#include <fstream>

#include "Players.h"

#include "Color.h"

#include "Winner.h"

using namespace std;

void Winner::AnnceWinner(int x){

winner = x;

ofstream output;

output.open("Game.txt",ios::app);

cout << "The winner is " << getPlayer(x) << endl;

output << "\nThe winner is " << getPlayer(x) << endl;

cout << "Congratulations!!!!" << endl;

output << "Congratulations!!!!" << endl;

}

GameBoard header and source

#ifndef GAMEBOARD\_H

#define GAMEBOARD\_H

#include "Color.h"

class GameBoard: public color{

private:

char Board;

public:

void initBoard();

char compareBoard();

void setBoard(char);

void displayBoard();

};

#endif /\* GAMEBOARD\_H \*/

#include <cstdlib>

#include <iostream>

#include <string>

#include <fstream>

#include "GameBoard.h"

#include "Color.h"

using namespace std;

void GameBoard::initBoard(){

Board='O';

}

char GameBoard::compareBoard(){

return Board;

}

void GameBoard::setBoard(char x){

Board = x;

}

Flowchart

