Project 1

<Connect 4>

CIS 17A

Dennis C Lang III

07/14/20

Introduction

Connect Four is a game played on a vertical game board that is six by seven units large. The game is designed for two players, and the objective is for one player to connect four of their game pieces either vertically, horizontally, or diagonally before the opponent is able to, and before the game board fills up entirely with pieces. After choosing which player goes first, each player drops a game piece (red or black in the physical game, 'X' or "O" in my game) into the column of choice. The piece then drops into the bottom-most row, stacking on top of prior pieces until the column is full. The physical game uses gravity and a shelf on the bottom row, as well as a very thin game board to achieve this, while the code uses a 2D array of empty tiles and counts upward from one, the bottom row. To restart the game, the board is wiped, and the players continue from scratch.

Summary

Lines of Code: ~300 including comments, ~270 without

Variables: around 12

Functions: about 20, including resets, re-assigning players, and building the game.

This program started with building the game board (the six by seven 2D array) and then creating the rules for the game. The game allows the users to name themselves, as well as change who goes first and who goes second before continuing with the game. Each user takes turns "dropping" their game piece into the board in their desired column, defaulting to the bottom-most row of course, with exception handling for a full column and choosing numbers outside of the 1-7 range. The program then increments each players' wins and losses after each match, writing and reading the results to the same BIN file each time.

Description

This program incorporates memory allocation with array and structures, functions with structures that input and output to the program, pointers with arrays and arrays of structures(used internally as well as externally), character arrays and string objects, and reading and writing to binary files to record the win and loss totals for each user and increment after each game played.

Using an array of structures containing two Players, the program assigns each player a string name and a game piece. Using functions and pointers, the data is referenced and kept in memory to allow each player to take turns and check if a win has been found. Scores are written to the BIN file and read out after each completed game. The option to reset after each completed game is presented (whether a winner is found or if the board is filled).

Pseudo Code

Ask for users' names

Check if names are correct, in the correct play order

If 'n', loop through until users verify input with 'y'

If other entry, loop through until users verify 'y' or'n'

Assign X and O to p1 and p2 respectively

Create game board of empty 6x7 2D array

Player 1 place game piece on 1-7, piece goes to lowest row in column

If piece is already in all 7 rows of the selected column, re-enter input

Player 2 same, repeat until..

While !win, continue filling up board

Check win for each piece played, give the win to owner of final game piece

Check win by incrementing +3 game pieces in any direction from previous piece played

When 1 user wins, p1.win++ or p2.win++

Print win and loss totals for each user from BIN

Ask to play again

If play again

Clear all values from the 2D array

Create game board of empty 6x7 array...

Connect Four Program

Dennis Lang
07/14/20

Create a 2D char array and have players take turns filling it, checking after each turn for 4 connected game pieces in either direction

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
```

```
Struct Stats{ int winTot,
    lossTot = 0;};
Struct player{
    string name;
    char XorO;
    Struct Stats getTots;};
enum { P1, P2 };
```

```
char reset (char sixBy7[6][7]);
int isFour (char sixBy7[6][7], Player current);
int takeTrn(char sixBy7[6][7], Player current);
void confirm(Player *ptr);
void getSide(Player *ptr);
void gmeBord (char sixBy7[6][7]);
void isValid (char sixBy7[6][7], Player current,
                int slction);
void playGme (char sixBy7[6][7], Player *ptr);
Player gtPlaDat();
```

A

A

Connect 4 Game

Declare Variables
Player play;
Player arr[2];
char sixBy7[6][7]

```
//cout << placeholder struct Player
//use of struct function return
play = gtPlaDat();
```

B

B

```
//get player names, assign sides
getSide(arr);
```

```
//ask if input is correct
confirm(arr);
```

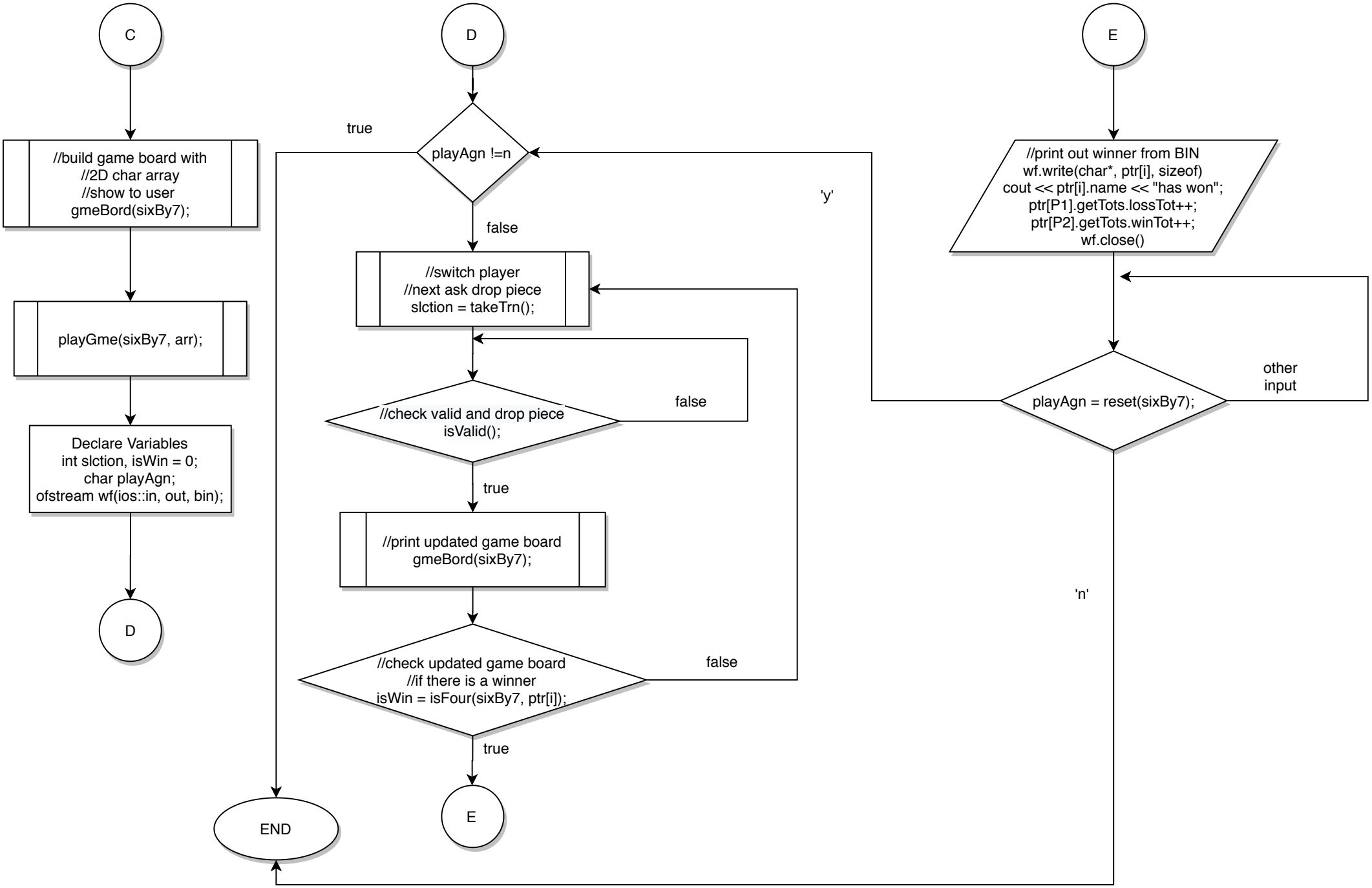
cin >> char conf;

conf = 'n'

conf != 'y' &&
conf != 'n'

conf = 'y'

C


C
//build game board with
//2D char array
//show to user
gmeBord(sixBy7);
playGme(sixBy7, arr);
Declare Variables
int slction, isWin = 0;
char playAgn;
ofstream wf(ios::in, out, bin);
D
D
true
playAgn !=n
false
//switch player
//next ask drop piece
slction = takeTrn();
//check valid and drop piece
isValid();
false
true
//print updated game board
gmeBord(sixBy7);
//check updated game board
//if there is a winner
isWin = isFour(sixBy7, ptr[i]);
false
true
E
END
E
//print out winner from BIN
wf.write(char*, ptr[i], sizeof)
cout << ptr[i].name << "has won";
ptr[P1].getTots.lossTot++;
ptr[P2].getTots.winTot++;
wf.close()
playAgn = reset(sixBy7);
other
input
'y'
'n'

Project Check Off Sheet

Ch	Sec	Concept		Location in Code	Comments
9		**Pointers/Memory Allocation**			
	1	Memory Addresses			
	2	Pointer Variables	5	void confirm(Player *ptr); //line 29	Store inputs in struct mem
	3	Arrays/Pointers	5	char sixBy7[7][7]; //line 41	Creates 2D array
	4	Pointer Arithmetic			
	5	Pointer Initialization			
	6	Comparing			
	7	Function Parameters	5	void confirm(Player *ptr); //line 29	Passes struct to confirm
	8	Memory Allocation	5	array = new int[num]; //line 279	Allows for dynamic
	9	Return Parameters	5	int *dynamic(int num); //line 34	Returns dynamic mem
	10	Smart Pointers			
10		**Char Arrays and Strings**			
	1	Testing			
	2	Case Conversion			
	3	C-Strings	10	char sixBy7[7][7]; //line 41	Creates 2D array (char)
	4	Library Functions			
	5	Conversion			
	6	Your own functions			
	7	Strings	10	string name; //line 19	Allows for first+last name
11		**Structured Data**			
	1	Abstract Data Types			
	2	Data			
	3	Access			
	4	Initialize			
	5	Arrays	5	Player arr[2]; //line 40	Array of 2 Player structs
	6	Nested	5	struct Stats getTots; //line 21	1st struct inside 2nd
	7	Function Arguments	5	void getSide(Player *ptr); //line 30	Struct as argument
	8	Function Return	5	Player play = gtPlaDat(); //line 43	Defines a temp player
	9	Pointers	5	(Player *ptr) //line 30	Struct as pointer
	10	Unions ****			
	11	Enumeration	5	enum {PLAYER1, PLAYER2}; //line 24	P1 = 0, P1 = 1
12		**Binary Files**			
	1	File Operations			
	2	Formatting	2	Setw(2) //line 253	Set weight for <100 games
	3	Function Parameters	2	Player gtPlaDat(fstream &file){ //line 267	Pass fstream obj by ref
	4	Error Testing			
	5	Member Functions	2	getline (cin,ptr[PLAYER1].name); //line 53	Allow whitespace in name
	6	Multiple Files	2	ofstream outFile("out.txt"); file.open("test.txt", ios::in); //line 270	Read from 1, write to 2
	7	Binary Files	5	ofstream wf("score.bin", ios::in | ios::out | ios::binary); //line 229	Write to BIN
	8	Records with Structures	5	wf.write((char *) &ptr[i], sizeof(&ptr[i])); //line 250	Struct is written (ptr)
	9	Random Access Files	5	file.seekg(11L, ios::beg); //272	Read starts at line 12
	10	Input/Output Simultaneous	2	ofstream wf("score.bin", ios::in | ios::out | ios::binary); //line 229	Read and write to same

```
/*
 * File: Connect Four Project
 * Author: Dennis Lang
 * Created on July 9, 2020, 11:13 PM
 */

#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
using namespace std;

struct Stats{
   unsigned int winTot = 0,
                lossTot = 0;
};

struct Player{
   string name;
   char XorO;
   struct Stats getTots;
};

enum {PLAYER1, PLAYER2};

char reset (char sixBy7[6][7], fstream &file);
int isFilld(char sixBy7[6][7]);
int isFour (char sixBy7[6][7], Player current);
int takeTrn(char sixBy7[6][7], Player current);
void confirm(Player *ptr);
void getSide(Player *ptr);
void gmeBord (char sixBy7[6][7]);
void isValid (char sixBy7[6][7], Player current, int slction);
void playGme (char sixBy7[6][7], Player *ptr);
int *dynamic(int num);
Player gtPlaDat(fstream &file);

int main(int argc, char** argv) {
   Player play;
   Player arr[2]; //create 2 players, p1 and p2
   char sixBy7[7][7]; //allow buffer for arr[1], '\n'?
   cout << "Welcome to Connect Four!" << endl;
   getSide(arr);
   confirm(arr);
   gmeBord(sixBy7);
   playGme(sixBy7, arr);
   return 0;
}

void getSide(Player *ptr){
   //get each side and then confirm();
   cout << "Player X will go 1st, enter your name: ";
   getline (cin,ptr[PLAYER1].name);
   cout << "Player O will go 2nd, enter your name: ";
   getline (cin,ptr[PLAYER2].name);
   ptr[PLAYER1].XorO = 'X';
   ptr[PLAYER2].XorO = 'O'; //assign X's and O's
   for(int i=0; i<2; i++){
   cout << ptr[i].name << " will be the "
        << ptr[i].XorO << "'s." << endl;
   }
}

void confirm(Player *ptr){
   char conf;
   cout << "Is this correct (y/n)?" << endl;
```

```
65      cin >> conf;
66      if(conf == 'n'){   //allows users to re-decide who goes first
67          cout << "Please try again." << endl;
68          cin.ignore();
69          getSide(ptr);
70          confirm(ptr);
71      }else if(conf != 'y' && conf != 'n'){
72          confirm(ptr);
73      }else{
74          cout << endl;
75      }
76      cout << "Get ready to play." << endl;
77      cout << "Enter any non-number to quit mid-game." << endl;
78      cout << "Connect four to win!" << endl;
79  }
80
81  int takeTrn( char sixBy7[6][7], Player current ){
82      //switch between players
83      //exception handling to not enter integer
84      //enter a non-numerical value to quit
85      int slction;
86      if(cin.fail()){
87          cout << "Exit game" << endl;
88          exit(0);
89      }
90      do{
91          if(!cin.fail()){
92              cout << current.name << "'s Turn ";
93              cout << "Enter a number (1-7): ";
94              cin >> slction;
95              if(!cin.fail()){
96                  while (sixBy7[1][slction] != ' '){
97                      cout << "Row Full, Choose 1-7: ";
98                      cin >> slction;
99                  }
100             }else{
101                 break;
102             }
103     }else{
104         break;
105     }
106     }while (slction < 1 || slction > 7);
107
108 return slction;
109 }
110
111 void isValid (char sixBy7[6][7], Player current, int slction){
112     //checks and prints value per each 2D array value
113     int SIZE, getTurn = 0;
114     SIZE = getTurn + 6;
115     while (getTurn != 1){
116         if (sixBy7[SIZE][slction] != 'X'
117             && sixBy7[SIZE][slction] != 'O'){
118             sixBy7[SIZE][slction] = current.XorO;
119             getTurn = 1;
120         }else{
121             --SIZE;
122         }
123     }
124 }
125
126 void gmeBord (char sixBy7[6][7]){
127     int array[6];
128 //print game board after each turn, verify is isFour, create array of 1-7 to label
129     for(int i=1; i<=6; i++){
130         for(int j=1; j<=7; j++){
131
```

```
            if(sixBy7[i][j] != 'O' && sixBy7[i][j] != 'X'){
                sixBy7[i][j] = ' ';
            }
            cout << sixBy7[i][j] << "|";
        }
        cout << endl;
    }

    for(int i=1; i<=7; i++){
        array[i] = i;
        cout << array[i] <<  " ";
    }
    cout << endl;
}

int isFour (char sixBy7[6][7], Player current){
    int win = 0;
    char c = current.XorO;

//check each position in the array of chars
//if X's or O's are 4 in a row in any direction
//print out win = 1 to the playGame function to proceed
    for (int i=1; i<4; i++){
        for (int j=1; j<=7; j++){
            if (sixBy7[i][j] == sixBy7[i + 1][j] &&
                sixBy7[i][j] == sixBy7[i + 2][j] &&
                sixBy7[i][j] == sixBy7[i + 3][j] &&
                sixBy7[i][j] == c){
                win = 1;
            }
        }
    }
    for (int i=1; i<7; i++){
        for (int j=1; j<=4; j++){
            if (sixBy7[i][j] == sixBy7[i][j + 1] &&
                sixBy7[i][j] == sixBy7[i][j + 2] &&
                sixBy7[i][j] == sixBy7[i][j + 3] &&
                sixBy7[i][j] == c){
                win = 1;
            }
        }
    }
    for (int i = 1; i<=3; i++){
        for (int j = 1; j<=4; j++){
            if (sixBy7[i][j] == sixBy7[i + 1][j + 1] &&
                sixBy7[i][j] == sixBy7[i + 2][j + 2] &&
                sixBy7[i][j] == sixBy7[i + 3][j + 3] &&
                sixBy7[i][j] == c){
                win = 1;
            }
        }
    }
    for (int i=0; i<4; i++) {
        for (int j=3; j<6; j++){
            if (sixBy7[i][j] == sixBy7[i + 1][j - 1] &&
                sixBy7[i][j] == sixBy7[i + 2][j - 2] &&
                sixBy7[i][j] == sixBy7[i + 3][j - 3] &&
                sixBy7[i][j] == c){
                win = 1;
            }
        }
    }
   return win;
}

char reset (char sixBy7[6][7], fstream &file){
```

```
    //verify 'y' or 'n' for replay
    Player play;
    int num = 6;
    char yOrN;
    if(yOrN != 'n'){
        cout << "Play again? (y/n) " << endl;
        cin  >> yOrN;
        if (yOrN == 'y'){
            for(int i=1; i<=6; i++){
                for(int j = 1; j <= 7; j++){
                    sixBy7[i][j] = ' ';  //clear sixBy7
                }
            }
            gmeBord(sixBy7);
        }else if (yOrN == 'n'){
            cout << "Good game(s)!" << endl;
            dynamic(num);
            play = gtPlaDat(file);
            cout << "Goodbye.";
            exit(0);
        }else{
            yOrN = reset(sixBy7, file);
        }
    }
    return yOrN;
}

void playGme (char sixBy7[6][7], Player *ptr){
    int slction, isWin, filled = 0;
    char playAgn;
    ofstream wf("score.bin", ios::in | ios::out | ios::binary);
//score.bin must exist
    fstream file;
//write to file the win and loss totals, increment and write each time
//loop depending on the play again variable returned by reset() after completion
    cout << "Rule: enter any non-integer to concede." << endl;
    while (playAgn != 'n'){
        for(int i=0; i<=1; i++){
            slction = takeTrn(sixBy7, ptr[i]);
            isValid(sixBy7,ptr[i], slction);
            gmeBord(sixBy7);
            isWin = isFour(sixBy7, ptr[i]);
            if(isWin == 1){
                ptr[i].getTots.winTot++;
                cout << ptr[i].name << " has won, congrats!" << endl;
                if(i == PLAYER1){
                    ptr[PLAYER2].getTots.lossTot++;
                }else{
                    ptr[PLAYER1].getTots.lossTot++;
                }
                for(int i=0; i<2; i++){
                    wf.write((char *) &ptr[i], sizeof(&ptr[i]));
                }
                for(int i=0; i<2; i++) {
                    cout << ptr[i].name << " has won "
                        << setw(2)
                        << ptr[i].getTots.winTot
                        << " time(s) and lost "
                        << setw(2)
                        << ptr[i].getTots.lossTot
                        << " time(s)."<< endl;
                }
                wf.close();
                playAgn = reset(sixBy7, file);
            }
            filled = isFilld(sixBy7);
```

```
            if (filled == 7){
                cout << "Tie game." << endl;
                playAgn = reset(sixBy7, file);
            }
        }
    }
}

Player gtPlaDat(fstream &file){
    char ch[20];
    ofstream outFile("out.txt");
    file.open("test.txt", ios::in);
    file.seekg(11L, ios::beg); //ignore programmer's score
    cout << endl << "Previous win streaks: " << endl;
    while (file >> ch[20]){
        outFile.put(ch[20]);
        cout << ch[20];
    }
    file.close();
    cout << endl << "This is the what the empty structure looks like:" << endl;
    Player temp;
    for(int i = 0; i < 2; i++){
        cout << "P" << i+1 << " = " << temp.name << endl;
        cout << "Game piece = " << temp.XorO << endl;
    }
    return temp;
}

int *dynamic(int num){
    //dynamic memory test
    int *array = nullptr;
    if(num <= 0)
        return nullptr;
    array = new int[num]; //allocate dynamic
    srand(time(0));
    cout << "Here's some lotto numbers:" << endl;
    for(int i =0; i < num; i++){
        array[i] = rand()%47;
        cout << array[i] << " ";
    }
    return array;
}

int isFilld(char sixBy7[6][7]){
//checks the array column by column to see if full board
    int filled = 0;
    for(int j = 1; j <= 7; j++){
        if(sixBy7[1][j] != ' '){
            ++filled;
        }
    }
    return filled;
}
```