Project I

Battleship

CIS-5-43518

May 14, 2022

Danielle Fernandez

# Table of Contents

[Introduction](#_7vuy6z8xeunm).…………………………………………..... 2

[Development Summary](#_773j3mf1d4gh)….……………………………... 2

[Sample Inputs/Outputs](#_yc1jru2tr54n)….…………………………….... 3

[Flowchart](#_qtq08qny3qkv)….……………………………………………. 5

[Psuedo Code](#_aebd2xmhlg4l) ………………………………………….. 11

[Major Variables](#_892pz4i5n333) ………………………………………. 13

[Checklist](#_4bt64bistp5e) …….……………………………………….. 14

[Reference](#_ka69mfukdv7n) ……….……………………………………..15

[Program](#_rw5khe90pw4y)……………………………………………….. 15

## 

## **Introduction**

This game simulates a slimmed-down version of the classic board game, Battleship.

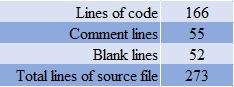
**Objective:**

Guess the location of your opponent's ship.

**Rules**:

* Each player has 1 ship.
* The board has 14 different locations that are represented by an integer between 1 and 14 (For ease of use, the computer automatically generates the ship’s location with a random number).
* The first player guesses a number 1-14. If they successfully guessed their opponent's ship location, then a “HIT” message is displayed. If their guess is wrong, then a “MISS” message is displayed and it is the other player’s turn.
* Players continue taking one guess at a time until someone gets a “HIT”.
  + It can sometimes take over 10 guesses before a player is correct, so for ease of use, the computer automatically generates each player’s guess with a random number.
* Once someone gets a win, a scoreboard will show how many wins each player has earned.
* The game will ask the user if they would like to play again.
* If they choose to play again, the game will restart. If the user chooses not to continue playing, then the game will show each player’s winning average and the average amount of rounds it takes to win 1 game.

## **Development Summary**



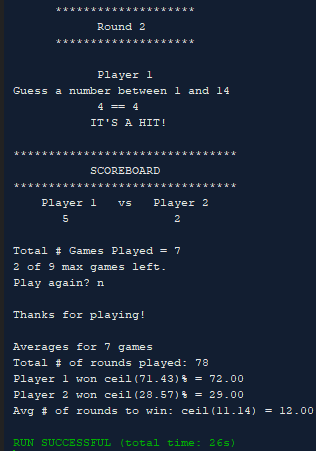
This game covers chapters 1 thru 5 in the [textbook](#_g76gu0qinzv5) by illustrating the 7 constructs, primitive data types, formatting, file i/o, etc…

My initial challenge was figuring out how to set the ships' locations without an array because traditionally the game’s board has 10 rows, and 10 columns and each ship holds anywhere from 2 to 5 pegs. I resolved this issue by simplifying the board to 2 rows with 7 columns which results in 14 numbers.

My second dilemma arose to whether or not to allow the user to make their own guesses. The problem is that while running it with a random function I could see it could sometimes take over 10 guesses each before someone got a hit. I am actually still debating writing a version 5 which would reduce the board to 7 numbers and allow a user to guess for themselves. The problem with that choice is that I have already filled out all my documentation, images, and flowcharts according to version 4. Version 4 fulfills all of the project's requirements without allowing a user to make their own guesses.

#### **Sample Inputs:** y Y y y y Y y N

#### **Sample Outputs:**



#### **Version 1**

This version 1 plays until a player wins 1 game and then asks if they want to play again. Ships' locations and players' guesses are randomly generated. I implemented a void banner() to display the game's name in a star banner. The game only allows the user to play up to 1 win before resetting game variables to their defaults.

#### **Version 2**

Inside the banner(string ) I added a for loop to output 3 rows, then there's a switch() that outputs the contents of each row. Rows 0 and 1 use a for loop to print 32 stars and row 1 prints the variable that was passed through its arguments.

#### **Version 3**

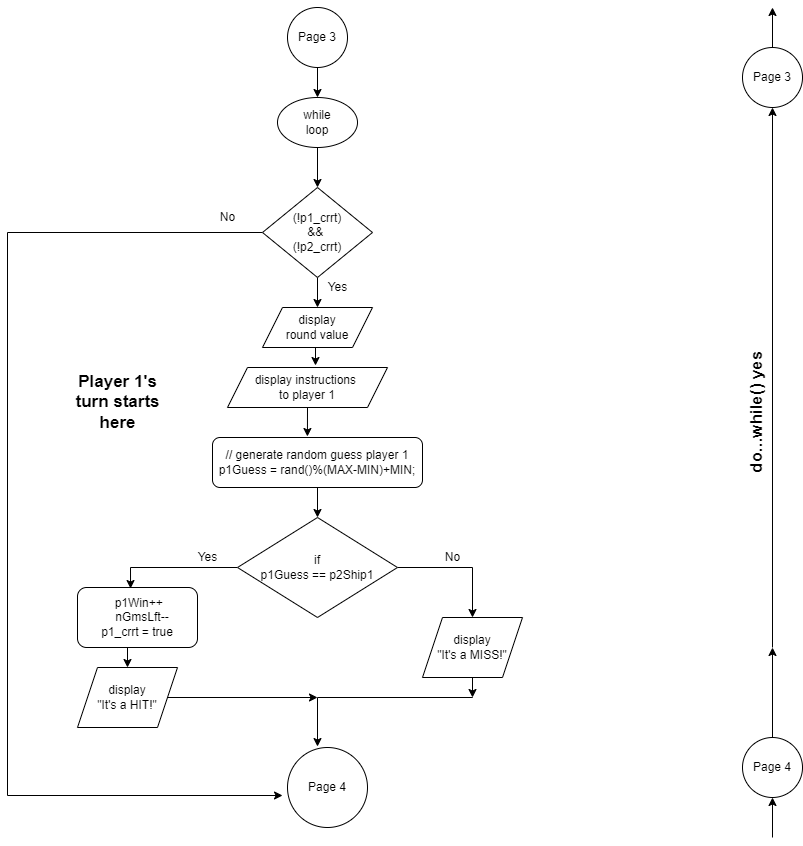
I added float variables, so I could find each player’s win average. I included the cmath library, but couldn’t get the round() to work. I added the fstream library and read in a file that held the max number of games and created an outFile that wrote the player's scores after each win to it. I removed the bool playAgn variable on line 98 in v2, and replaced it with 2 new bool variables (p1\_crrt, p2\_crrt). I used these as flags, see lines 100 & 129.This enabled me to change my while() which only allowed the game to play up to 1 game at a time.

#### **Version 4**

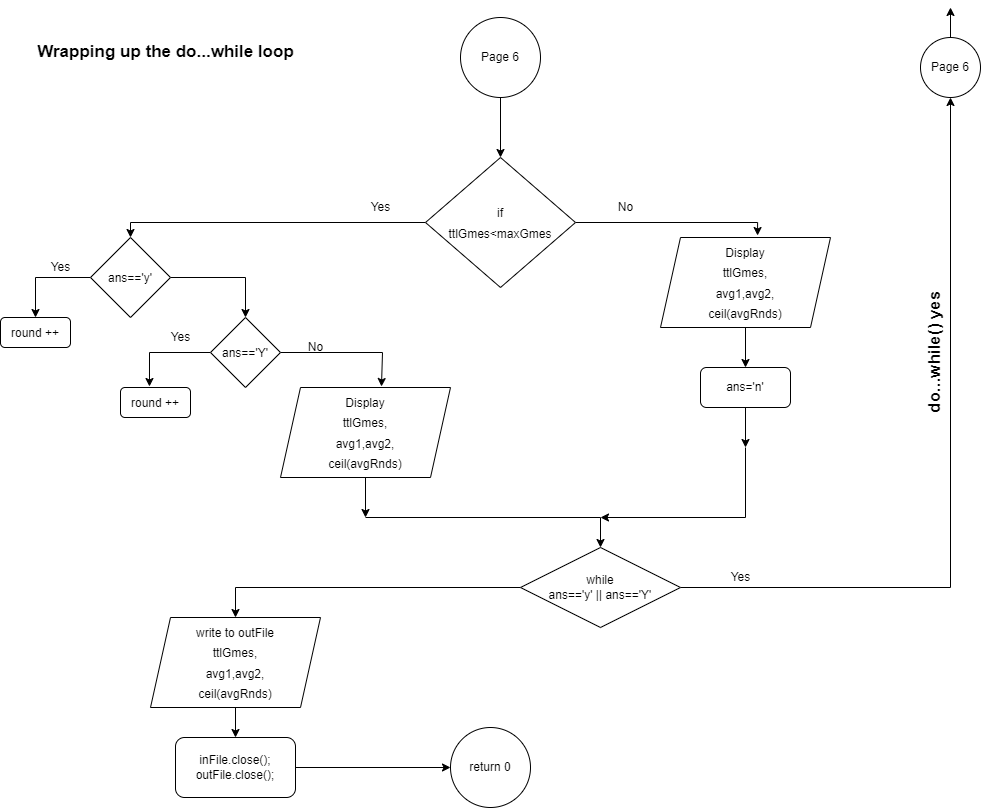
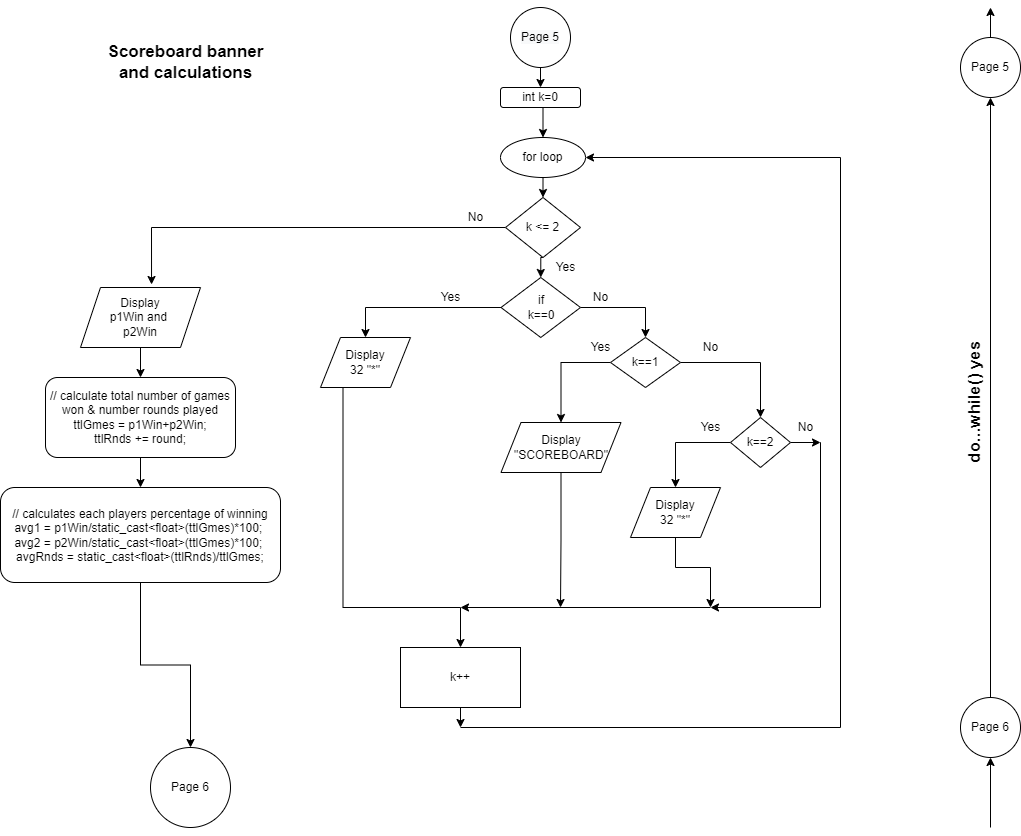
I removed my function that displayed a star banner with the string that was passed into it. This allowed me to use a ternary operator when I displayed the Scoreboard banner. Next, I created a total round variable to find the sum of the rounds and then found the average amount of rounds it took before a player got a hit. I did this because it sometimes took rounds to get a win, and I thought the ceil function would be more useful on it. Ultimately, I ended using the ceil() on each player’s average win as well as the total rounds because sometimes one or two of the variables didn’t need to be rounded up. Basically my odds of needing to round up were better if I used it 3 different times. I wasn’t sure if the checklist wanted an increment or a decrement, so I added a new variable nGmsLft (number of games left), so I could illustrate a decrement. I reassigned the ans variable to ‘n’ (line 243) which fixed my bug and ended the do..while loop that controls the entire game.

## **Flowchart**

## 



## 



## 

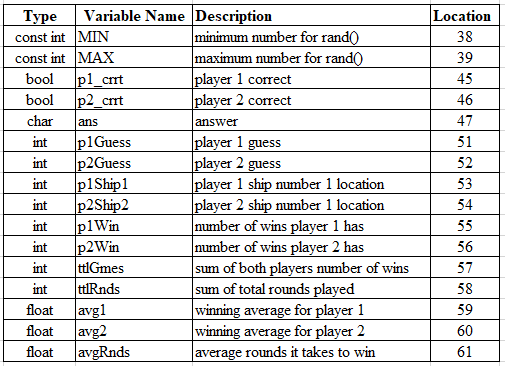
## **Pseudo Code**

* Include libraries: iostream, iomanip, cmath, cstdlib, fstream, cstring, ctime.
* Start main function.
* Set random number seed.
* Declare ifstream and ofstream variables for reading and writing files.
* Declare and initialize constant, bool, char, integer, float, string variables.
* Open inFile to that already exists with data in it.
* Create an outFile to write player’s final scores to.
* Read inFile integer that represents max number of games (maxGames) a user can play.
* Assign numGamesLeft to equal maxGames.
* do loop
  + Initialize each player's ship to random number between 1-14 to represent its location on game board
  + Set p1\_correct and p2\_correct to default starting values == false
  + for loop: display a 3-line game banner. for(int i=0; i<=2; i++)
    - * switch (i)
      * case 0:
      * case 2: For loop prints a line of 32 stars
      * case 1: Display “BATTLESHIP”
      * default: Display "Error in game banner”
  + Display games introduction message
  + while(p1\_correct && p2\_correct==false): loops until a player correctly guesses

opponents ship location.

* + - Display value for round variable
    - Display instructions for player 1
    - Automatically generate p1Guess to a rand num between 1-14
    - if conditional to check if p1Guess equals the location of p2Ship1
      * Increment p1win by 1
      * Decrement nGmsLft by 1
      * Reassign p1\_correct to true for correct guess
      * Display “It’s a HIT!”
    - else p1Guess is wrong
      * Display “It’s a MISS!”
    - if p1\_correct equals false then player 2’s turn begins
      * Display instructions for player 2
      * Automatically generate p2Guess to a rand num between 1-14
      * if conditional to check if p2Guess equals the location of p1Ship1
        + Increment p2win by 1
        + Decrement nGmsLft by 1
        + Reassign p2\_correct to true for correct guess
        + Display “It’s a HIT!”
      * else p1Guess is wrong
        + Display “It’s a MISS!”
      * Increment round variable by 1
      * if player 1 and player 2 both guessed wrong, then
        + Display "You Both Missed. Try Again…”
  + for loop to display a 3-line scoreboard banner. for(int k=0; k<=2; k++)
    - Ternary Operator: if k==0)||(k==2)
      * Display "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"
    - else if k==1, then
      * Display “SCOREBOARD”
    - else Display "Error in scoreboard banner”
  + Display player1 and player2’s number of wins
  + Calculate running total number of games won by both players
  + Calculate running total number of rounds played for each game
  + Calculate each players average percentage of winning
  + Calculate the average amount of rounds played to get a win
  + if conditional checks maximum number of games has NOT been reached
    - Display totalNumGames, numGamesLeft, maxGames
    - Display message asking user if they want to play again
    - Input user’s answer
    - if/else if/else condition to validate user’s input
      * if ans==y, then
        + Reset round variable to equal 1
      * else if ans==Y, then
        + Reset round variable to equal 1
      * else
        + Display “Thanks for playing”
        + Display win and round averages
        + Display rounded up averages with ceil( )
  + else
    - Display “Thanks for playing”
    - Display win and round averages
    - Display rounded up averages with ceil( )
    - Reassign answer variable to ‘n’
* End do…while(ans=='y')||(ans=='Y')
* Write each player’s wins, averages, maxGames, totalGames, totalRounds to outFile
* Close inFile that was being read
* Close outFile that scores and averages were written to
* return 0

## **Major Variables**



#### **Checklist**

## **Reference**

1. Gaddis, Tony. *Starting out with C++: From Control Structures through Objects*. 9th ed., Pearson, 2018.
2. Lehr, Mark. “2022\_Spring\_CSC\_CIS\_5/Projects at Master · ml1150258/2022\_spring\_csc\_cis\_5.” *GitHub*, 2022, https://github.com/ml1150258/2022\_Spring\_CSC\_CIS\_5/tree/master/Projects.

## **Program**

/\*

\* File: main.cpp

\* Author: Danielle Fernandez

\* Created on May 12, 2022, 5:16 PM

\* Purpose: Project 1. Covers chapters 1-5 in Gaddis. Battleship v3

\* Version 4: added ternary, removed banner(), added ceil(), added decrement

\*/

// System Libraries:

#include <iostream> // cin, cout

#include <iomanip> // fixed, setprecision()

#include <cmath> // round()

#include <cstdlib> // rand()

#include <fstream> // ofstream

#include <cstring> // string library

#include <ctime> // time library for rand()

using namespace std;

// User libraries

// Global Constants

// Physics/Chemistry/Math/Conversions

// Function prototypes

// Program execution begins here

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

// set random number seed

srand(static\_cast<unsigned int> (time(0)));

// declare variables

ifstream inFile; // for reading an existing file

ofstream outFile; // for outputting to a file

const int MIN = 1, // minimum number for rand()

MAX = 14; // maximum number for rand()

const string HIT = "IT\'S A HIT!\n";

const string MISS = "IT\'S A MISS!\n";

const string P1 = "Player 1 ";

const string P2 = "Player 2 ";

bool p1\_crrt, // player 1 correct

p2\_crrt; // player 2 correct

char ans; // answer

int maxGmes = 0, // number of games

nGmsLft, // number of games left

round = 1, // round

p1Guess, // player 1 guess

p2Guess, // player 2 guess

p1Ship1, // player 1 ship number 1

p2Ship1, // player 2 ship number 1

p1Win = 0, // number of wins player 1 has

p2Win = 0, // number of wins player 2 has

ttlGmes = 0,

ttlRnds = 0;

float avg1, // average number of wins for player 1

avg2, // average number of wins for player 2

avgRnds;

string gussMsg = "\nGuess a number between 1 and ";

// open an existing file that holds max number of games a user can play

inFile.open("maxNGms.txt");

// create a file to output to

outFile.open("scores.txt");

// read in maximum number of games that can be played from file

inFile >> maxGmes;

nGmsLft = maxGmes; // set numberOfGamesLeft to equal maxGames

do { // game starts here

// initial variables to represent the location of each player's ship

p1Ship1 = rand()%(MAX-MIN)+MIN;

p2Ship1 = rand()%(MAX-MIN)+MIN;

// sets variables to default starting values

p1\_crrt = p2\_crrt = false;

// display 3 line game banner

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

switch(i){ // print a border or string depending on i

case 0:

case 2:

{

for (int j=0; j<32; j++) {

cout << "\*";

}

cout << endl;

break;

}

case 1:

{

cout << setw(21) << "BATTLESHIP" << endl;

break;

}

default: cout << "Error in game banner.\n";

}

}

// display games introduction message

cout << setw(2) << " " << "Try to guess the location of \n"

<< setw(6) << " " << "the computer\'s ship." << endl;

// loops until a player correctly guesses opponents ship location

while((!p1\_crrt) && (!p2\_crrt)){

cout << endl << setw(26) << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << setw(18) << "Round " << round << endl;

cout << setw(26) << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Player 1's Guess \*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// display instructions to player 1

cout << endl << setw(21) << P1 << gussMsg << MAX << endl;

// program generates random number guess

p1Guess = rand()%(MAX-MIN)+MIN;

// checks if player1 guess is correct

if(p1Guess == p2Ship1){

// increment player 1 number of wins

p1Win++;

nGmsLft--; // decrease number of total games

// reassign player 1's value for a correct guess

p1\_crrt = true;

// display HIT message for correct guess

cout << setw(13) << p1Guess << " == " << p2Ship1 << endl;

cout << setw(23) << HIT << endl;

} else { // if player1 guess is wrong

// display MISS message for wrong guess

cout << p1Guess << endl;

cout << setw(23) << MISS;

}

// conditional only runs if player 1 misses player 2's ship

if(!p1\_crrt){

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Player 2's Guess \*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// display instructions to player 2

cout << endl << setw(21) << P2 << gussMsg << MAX << endl;

// program automatically generates a guess for player 2

p2Guess = rand()%(MAX-MIN)+MIN;

// conditional checks if guess is correct

if(p2Guess == p1Ship1){

// increment player 2 number of wins

p2Win++;

nGmsLft--; // decrease number of total games

// reassign player 2's value for a correct guess

p2\_crrt = true;

// display HIT message for correct guess

cout << setw(13) << p2Guess << " == " << p1Ship1 << endl;

cout << setw(23) << HIT << endl;

} else { // display message for wrong guess

cout << p2Guess << endl;

cout << setw(23) << MISS;

}

}

round++;

// if both players guess wrong, then increment round by 1

// and display message to tell them to continue guessing

if((!p1\_crrt) && (!p2\_crrt)){

cout << endl << "You Both Missed. Try Again...\n\n";

}

}// ends while()

// Display scoreboard banner

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

((k==0)||(k==2)) ? cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

: (k==1) ? cout << setw(21) << "SCOREBOARD" << endl

: cout << "Error in scoreboard banner.\n";

}

cout << setw(4) << " " << P1 << setw(4) << "vs" << setw(3) << " "

<< right << P2 << endl;

cout << setw(8) << p1Win << setw(16) << p2Win << endl;

// calculate total number of games won & number rounds played

ttlGmes = p1Win+p2Win;

ttlRnds += round; // sums the total number of rounds from all games

// calculates each players percentage of winning

avg1 = p1Win/static\_cast<float>(ttlGmes)\*100;

avg2 = p2Win/static\_cast<float>(ttlGmes)\*100;

avgRnds = static\_cast<float>(ttlRnds)/ttlGmes;

// checks maximum number of games has NOT been played

if(ttlGmes<maxGmes){

cout << "\nTotal # Games Played = " << ttlGmes << endl;

cout << nGmsLft << " of " << maxGmes << " max games allowed.\n";

cout << "Play again? ";

cin >> ans;

// conditional validates user's input

if(ans=='y'){

round = 1;

cout << endl << endl;

} else if(ans=='Y'){

round = 1;

cout << endl << endl;

} else {

cout << "\nThanks for playing!\n";

cout << fixed << showpoint << setprecision(2);

cout << "\nAverages for " << ttlGmes << " games \n"

<< "Player 1 won ceil(" << avg1 << ")% = "

<< ceil(avg1) << endl

<< "Player 2 won ceil(" << avg2 << ")% = "

<< ceil(avg2) << endl;

cout << "Avg # of rounds to win: ceil(" << avgRnds << ") = "

<< ceil(avgRnds) << endl;

}

} else { // display end of game results

cout << "Max number of games has been reached." << endl;

cout << fixed << showpoint << setprecision(2);

cout << "\nAverages for " << ttlGmes << " games \n"

<< "Player 1 won ceil(" << avg1 << ")% = "

<< ceil(avg1) << endl

<< "Player 2 won ceil(" << avg2 << ")% = "

<< ceil(avg2) << endl;

cout << "Avg # of rounds to win: ceil(" << avgRnds << ") = "

<< ceil(avgRnds) << endl;

// reassign ans so it will end the do..while()

ans = 'n';

}

// continue doing all the statements above until

// ans does not equal y or Y

} while((ans=='y')||(ans=='Y'));

// write scores and averages to file

outFile << fixed << showpoint << setprecision(2);

outFile << "Player 1 wins: " << p1Win << endl

<< "Player 2 wins: " << p2Win << endl

<< ttlGmes << " of " << maxGmes << " max games were played.\n"

<< "Total # of rounds played: " << ttlRnds << endl

<< "\nAverages for " << ttlGmes << " games" << endl

<< "Player 1 won ceil(" << avg1 << ")% = "

<< ceil(avg1) << endl

<< "Player 2 won ceil(" << avg2 << ")% = "

<< ceil(avg2) << endl

<< "Avg # of rounds to win: ceil(" << avgRnds << ") = "

<< ceil(avgRnds) << endl;

// close file being read in

inFile.close();

// close scores.txt file

outFile.close();

// exit code

return 0;

}