Project I

Battleship

CIS-5-43518

May 14, 2022

Danielle Fernandez

Table of Contents

Introduction

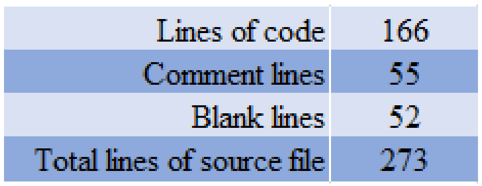
This game simulates a slimmed-down version of the classic board game, Battleship.

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



Lines of code	166
Comment lines	55
Blank lines	52
Total lines of source file	273

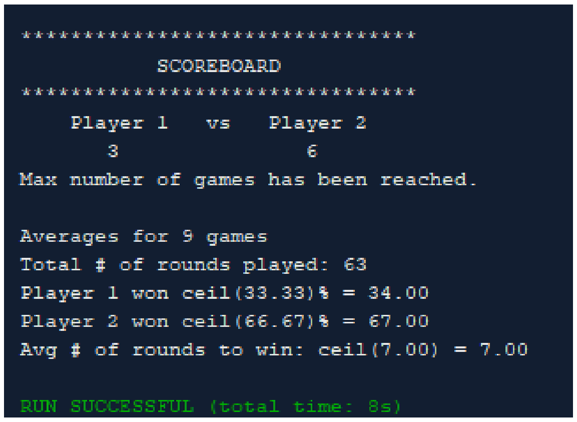
This game covers chapters 1 thru 5 in the textbook by illustrating the 7 constructs, primitive data types, formatting, file i/o, etc…

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

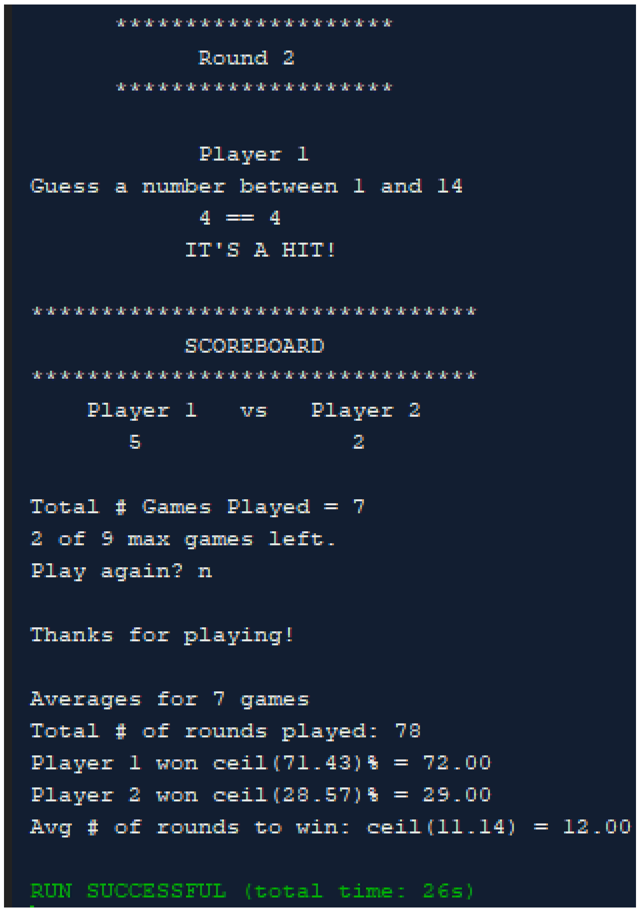
Sample Outputs:



```
********************************
           SCOREBOARD
********************************
    Player 1    vs    Player 2
       3               6
Max number of games has been reached.

Averages for 9 games
Total # of rounds played: 63
Player 1 won ceil(33.33)% = 34.00
Player 2 won ceil(66.67)% = 67.00
Avg # of rounds to win: ceil(7.00) = 7.00

RUN SUCCESSFUL (total time: 8s)
```



```
        ********************
              Round 2
        ********************


              Player 1
Guess a number between 1 and 14
              4 == 4
             IT'S A HIT!

********************************
           SCOREBOARD
********************************
    Player 1    vs    Player 2
       5               2

Total # Games Played = 7
2 of 9 max games left.
Play again? n

Thanks for playing!

Averages for 7 games
Total # of rounds played: 78
Player 1 won ceil(71.43)% = 72.00
Player 2 won ceil(28.57)% = 29.00
Avg # of rounds to win: ceil(11.14) = 12.00

RUN SUCCESSFUL (total time: 26s)
```

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

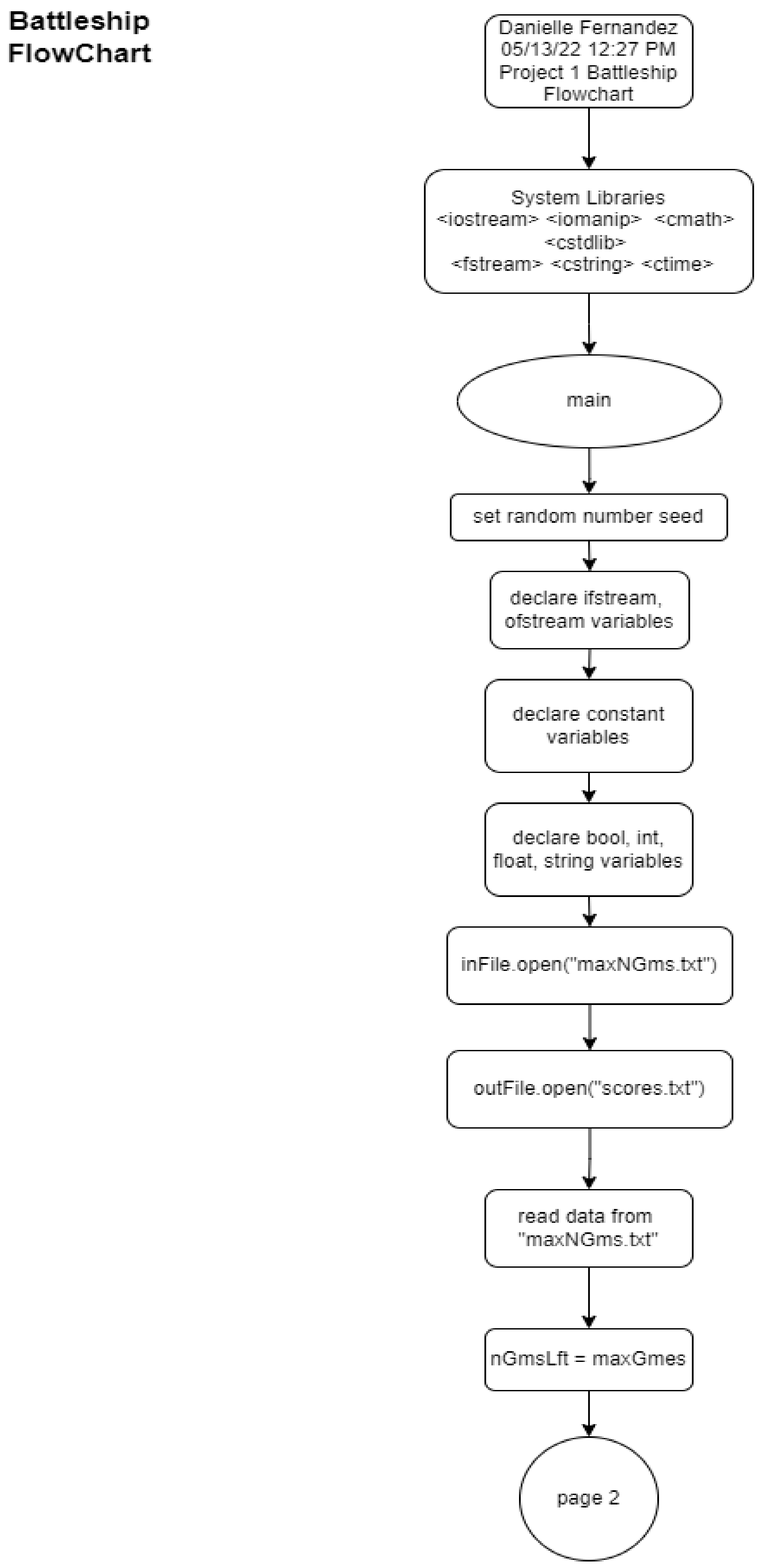
Inside the banner(string) I added a for loop to output 3 rows, then there's a switch() that outputs the contents of each row. Rows 0 and 1 use a for loop to print 32 stars and row 1 prints the variable that was passed through its arguments.

Version 3

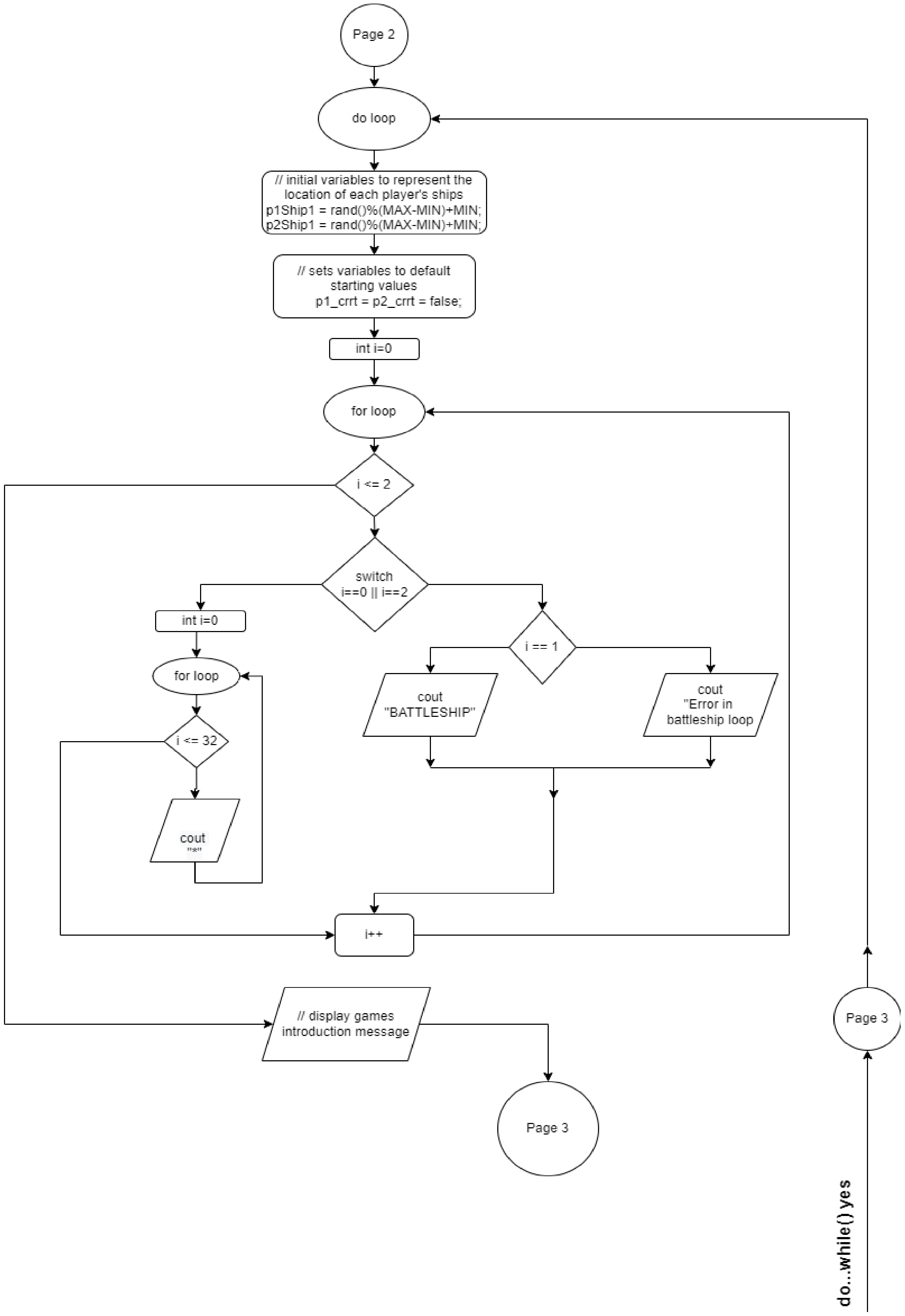
I added float variables, so I could find each player's win average. I included the cmath library, but couldn't get the round() to work. I added the fstream library and read in a file that held the max number of games and created an outFile that wrote the player's scores after each win to it. I removed 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.

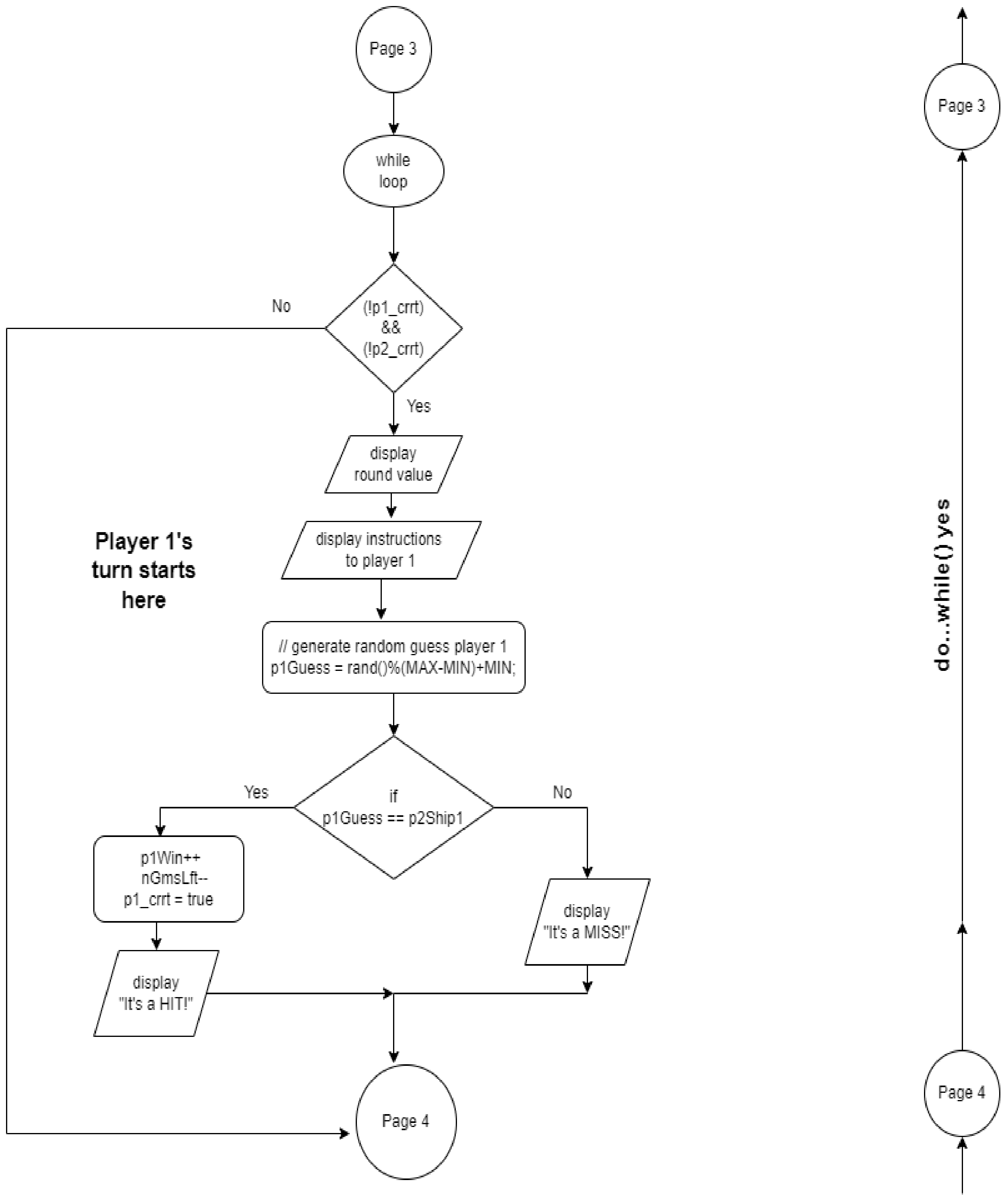
Version 4

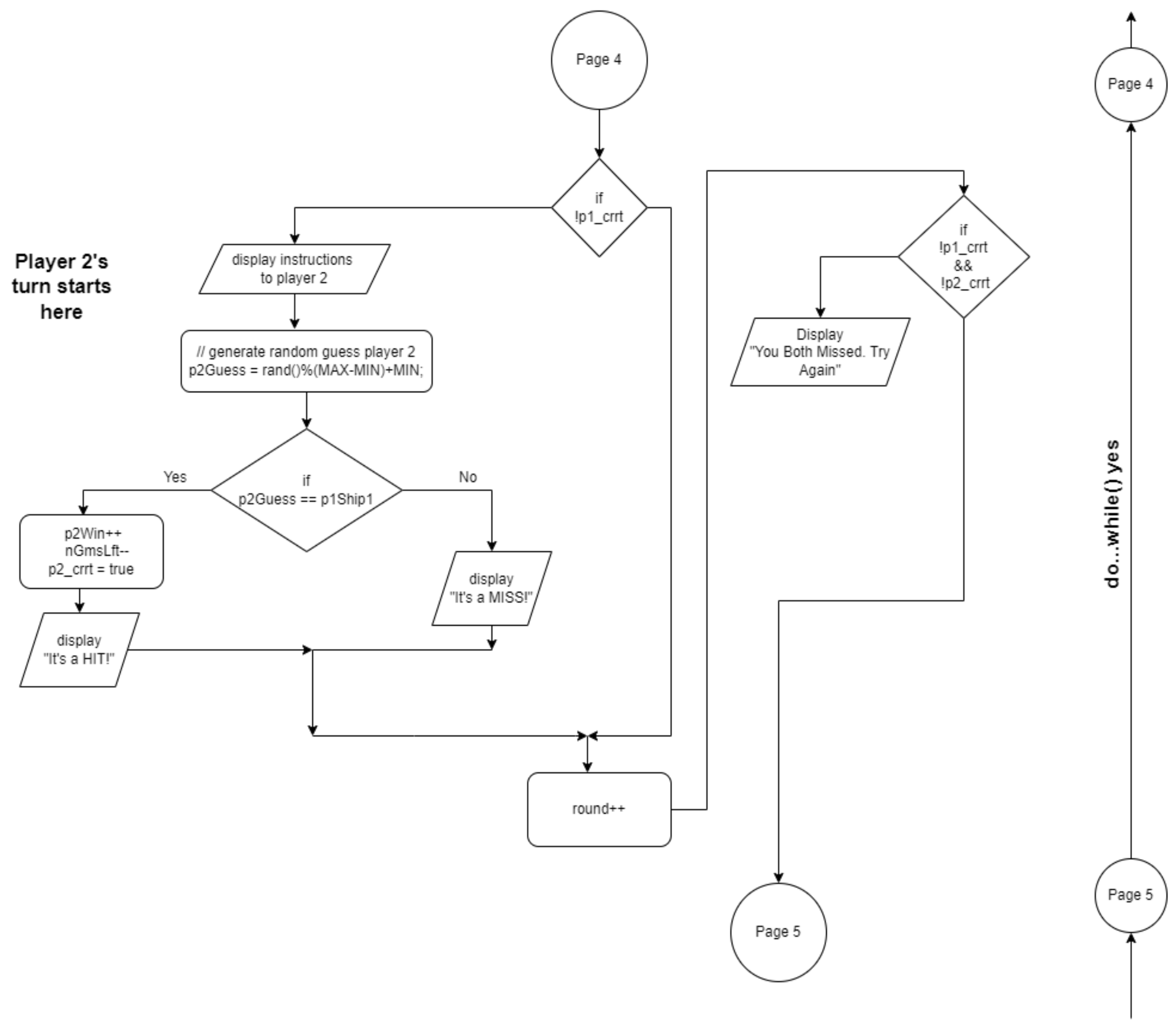
I removed my function that displayed a star banner with the string that was passed into it. This allowed me to use a ternary operator when I displayed the Scoreboard banner. Next, I created a total round variable to find the sum of the rounds and then found the average amount of rounds it took before 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.

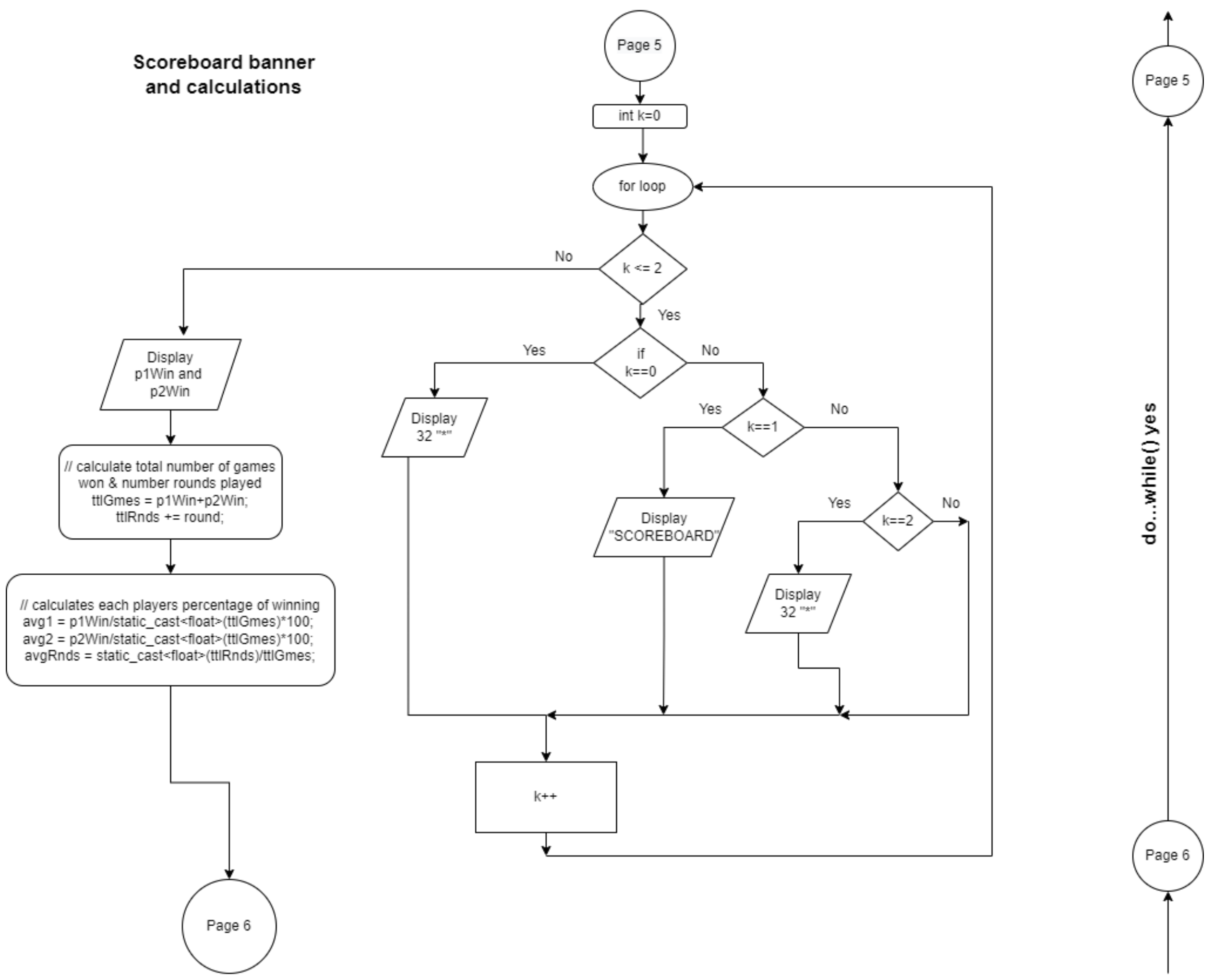

Battleship
FlowChart
Danielle Fernandez
05/13/22 12:27 PM
Project 1 Battleship Flowchart
System Libraries
<iostream> <iomanip> <cmath> <cstdlib>
<fstream> <cstring> <ctime>
main
set random number seed
declare ifstream, ofstream variables
declare constant variables
declare bool, int, float, string variables
inFile.open("maxNGms.txt")
outFile.open("scores.txt")
read data from "maxNGms.txt"
nGmsLft = maxGmes
page 2

Flowchart

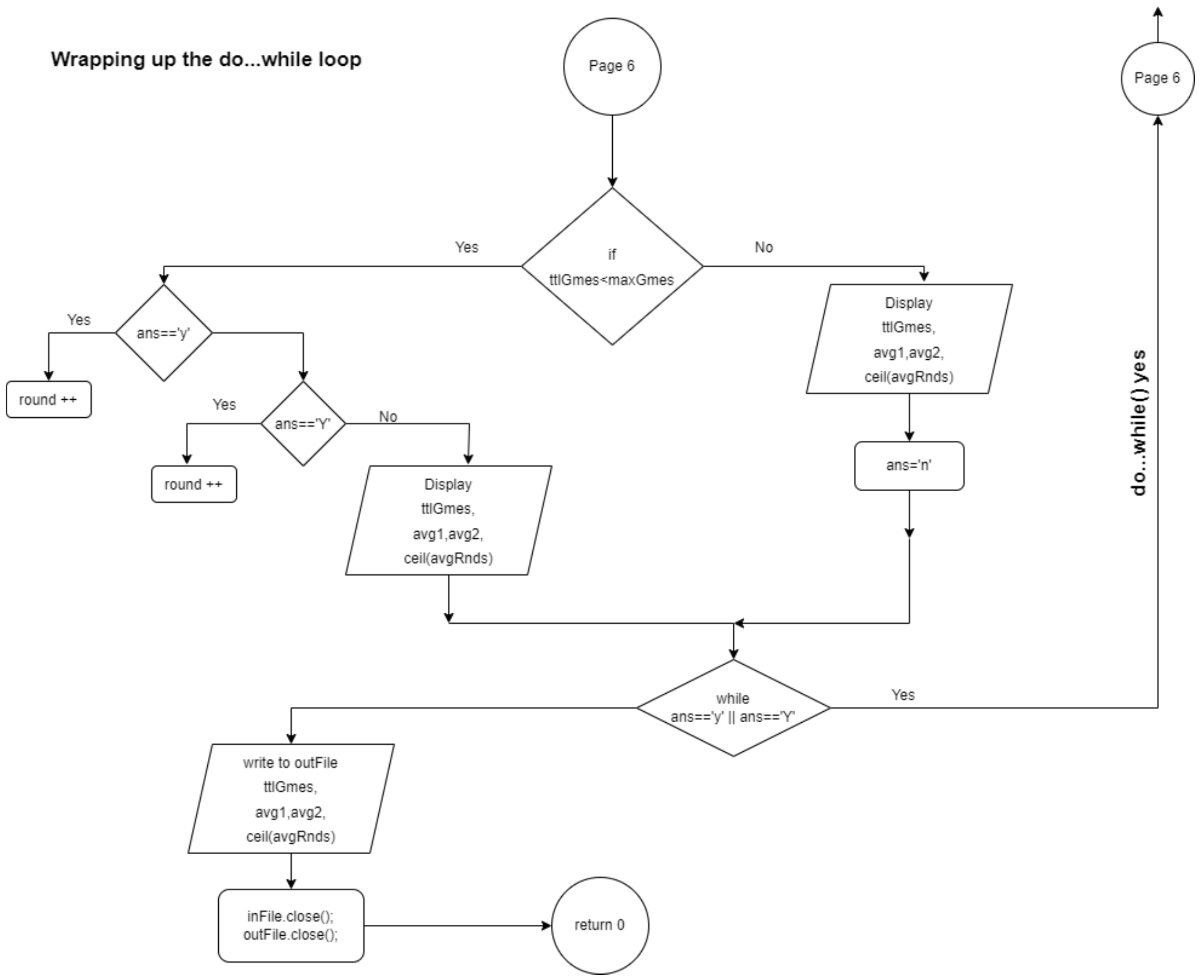

Page 2
do loop
// initial variables to represent the location of each player's ships
p1Ship1 = rand()%(MAX-MIN)+MIN;
p2Ship1 = rand()%(MAX-MIN)+MIN;
// sets variables to default starting values
p1_crrt = p2_crrt = false;
int i=0
for loop
i <= 2
switch
i==0 || i==2
int i=0
for loop
i <= 32
cout
"*"
i == 1
cout
"BATTLESHIP"
cout
"Error in battleship loop
i++
// display games introduction message
Page 3
Page 3
do..while() yes


Page 3
while loop
(!p1_crrt) && (!p2_crrt)
No
Yes
display round value
display instructions to player 1
Player 1's turn starts here
// generate random guess player 1
p1Guess = rand()%(MAX-MIN)+MIN;
if p1Guess == p2Ship1
Yes
No
p1Win++
nGmsLft--
p1_crrt = true
display "It's a HIT!"
display "It's a MISS!"
Page 4
Page 3
do...while() yes
Page 4


Player 2's turn starts here
Page 4
if !p1_crrt
display instructions to player 2
// generate random guess player 2
p2Guess = rand()%(MAX-MIN)+MIN;
if p2Guess == p1Ship1
Yes
No
p2Win++
nGmsLft--
p2_crrt = true
display "It's a HIT!"
display "It's a MISS!"
round++
if !p1_crrt && !p2_crrt
Display "You Both Missed. Try Again"
Page 5
Page 4
do...while() yes
Page 5


Scoreboard banner and calculations
Page 5
int k=0
for loop
k <= 2
No
Yes
if k==0
Yes
No
Display p1Win and p2Win
// calculate total number of games won & number rounds played
ttlGmes = p1Win+p2Win;
ttlRnds += round;
// 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;
Page 6
Display 32 "*"
k==1
Display "SCOREBOARD"
k==2
Display 32 "*"
k++
Page 5
do...while() yes
Page 6

Wrapping up the do...while loop



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

Type	Variable Name	Description	Location
const int	MIN	minimum number for rand()	38
const int	MAX	maximum number for rand()	39
bool	p1_crrt	player 1 correct	45
bool	p2_crrt	player 2 correct	46
char	ans	answer	47
int	p1Guess	player 1 guess	51
int	p2Guess	player 2 guess	52
int	p1Ship1	player 1 ship number 1 location	53
int	p2Ship2	player 2 ship number 1 location	54
int	p1Win	number of wins player 1 has	55
int	p2Win	number of wins player 2 has	56
int	ttlGmes	sum of both players number of wins	57
int	ttlRnds	sum of total rounds played	58
float	avg1	winning average for player 1	59
float	avg2	winning average for player 2	60
float	avgRnds	average rounds it takes to win	61

Checklist

Chapter	Section	Topic	Where Line #"s	Pts	Notes
2	2	cout			
	3	libraries	10-16	8	iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals			No variables in global area, failed project!
	5	Identifiers			
	6	Integers	48	3	
	7	Characters	47	3	
	8	Strings	62, 118	3	
	9	Floats No Doubles	59	3	Using doubles will fail the project, floats OK!
	10	Bools	45, 131	4	
	11	Sizeof *****			
	12	Variables 7 characters or less	x		All variables <= 7 characters
	13	Scope ***** No Global Variables	x		
	14	Arithmetic operators	121		
	15	Comments 20%+	x	5	Model as pseudo code
	16	Named Constants	38		All Local, only Conversions/Physics/Math in Global area
	17	Programming Style ***** Emulate	x		Emulate style in book/in class repository
3	1	cin	209		
	2	Math Expression	195		
	3	Mixing data types ****	199		
	4	Overflow/Underflow ****			
	5	Type Casting	199	4	
	6	Multiple assignment *****	81		
	7	Formatting output	190, 220	4	
	8	Strings	167	3	
	9	Math Library	224	4	All libraries included have to be used
	10	Hand tracing ******			

4	1	Relational Operators	124		>,<, ==
	2	if	178	4	Independent if
	4	If-else	156-169	4	
	5	Nesting	145-173	4	
	6	If-else-if	212-218	4	
	7	Flags *****	163		
	8	Logical operators	109, 249	4	&& || !NOT
	11	Validating user input	212-218	4	
	13	Conditional Operator	185	4	ternary
	14	Switch	85	4	
5	1	Increment/Decrement	127, 128	4	
	2	While	109	4	
	5	Do-while	74-249	4	
	6	For loop	184	4	
	11	Files input/output both	65,68,253	8	
	12	No breaks in loops ******	x		Failed Project if included
****** Not required to show			Total	100	

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;
}
```