**BattleShip game**

**Will Gunadi**

**CSC-5 44188**

**Spring 2017**

**Final Project**

**Table of Content**

**Intro**

**How the game works**

**Approach to the game**

**Flowchart**

**Construct/Concept Checklist**

**Refrences**

**Program**

**Introduction:**

Battleship is a [guessing game](https://en.wikipedia.org/wiki/Guessing_game) for two players. It is played on ruled grids (paper or board) on which the players' fleets of ships (including battleships) are marked. The locations of the fleet are concealed from the other player. Players alternate turns calling "shots" at the other player's ships, and the objective of the game is to destroy the opposing player's fleet. Both players take turns choosing coordinate of where the opponent ships are located. If the coordinates are correct, the ship in that location is sunk and gets another crack at sinking the other ships. The game is fairly simple but does require the ability to guess where the ships are, so luck is a big part of it.

**How the game works**

**Objective:** Sink all of the ships that are located.

Rules: Players take turn choosing numbers representing the x and y coordinates. If there is a ship on the coordinates, then that ship is sunk/damaged. If player hits, they keep trying until they miss. Once all of the ships are sunk, the game ends and the first player to do so, wins.

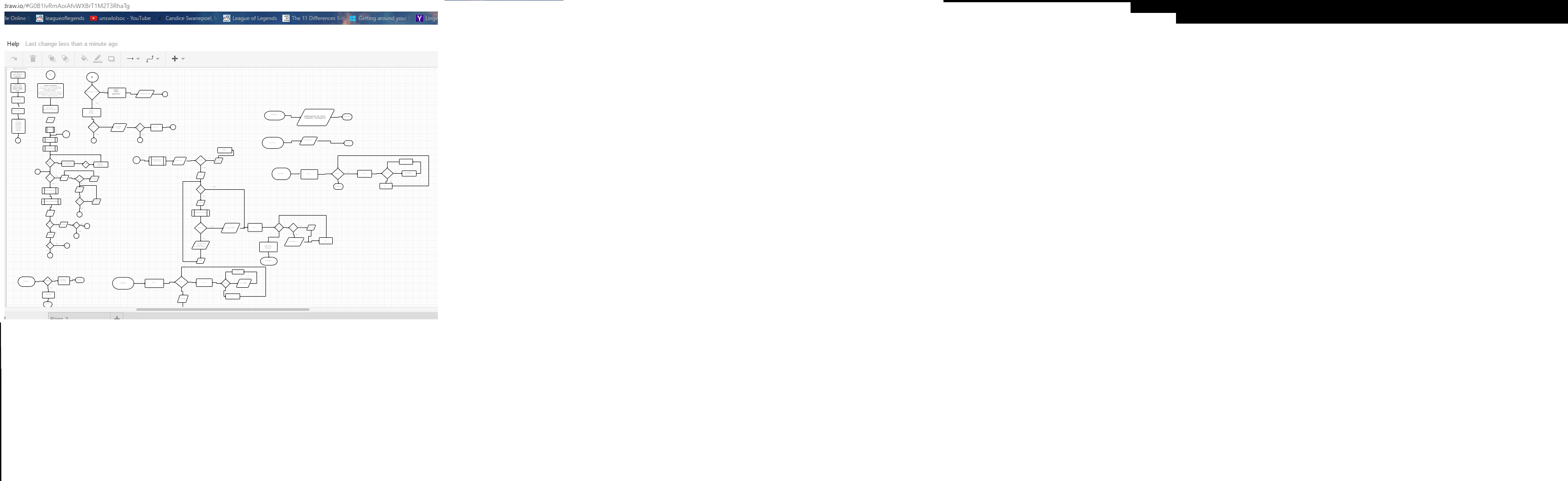
**Approach to the game**

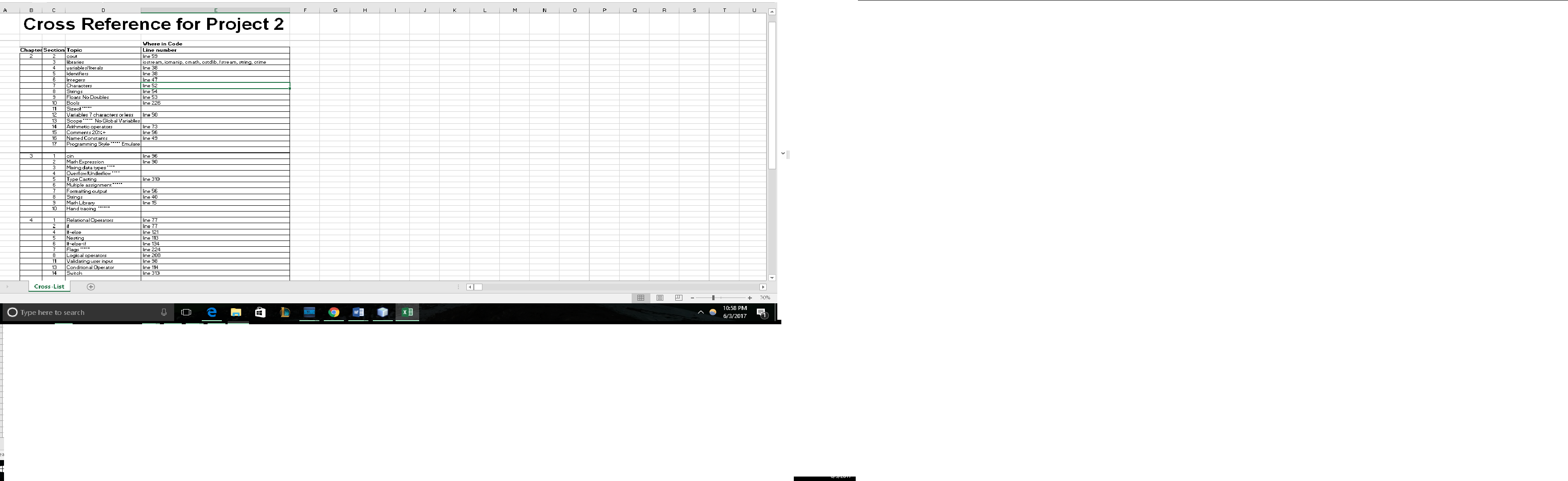
Usually, this game is played against another player or a computer. Because my current knowledge isn’t vast enough to create a game with another player or a computer, I’ve simplified the game to make a battleship sinking game. My game has the basic framework of the original game where player 1 chooses coordinates to find out whether or not there’s a ship at that location. If all n Ships are sunk, player 1 wins.

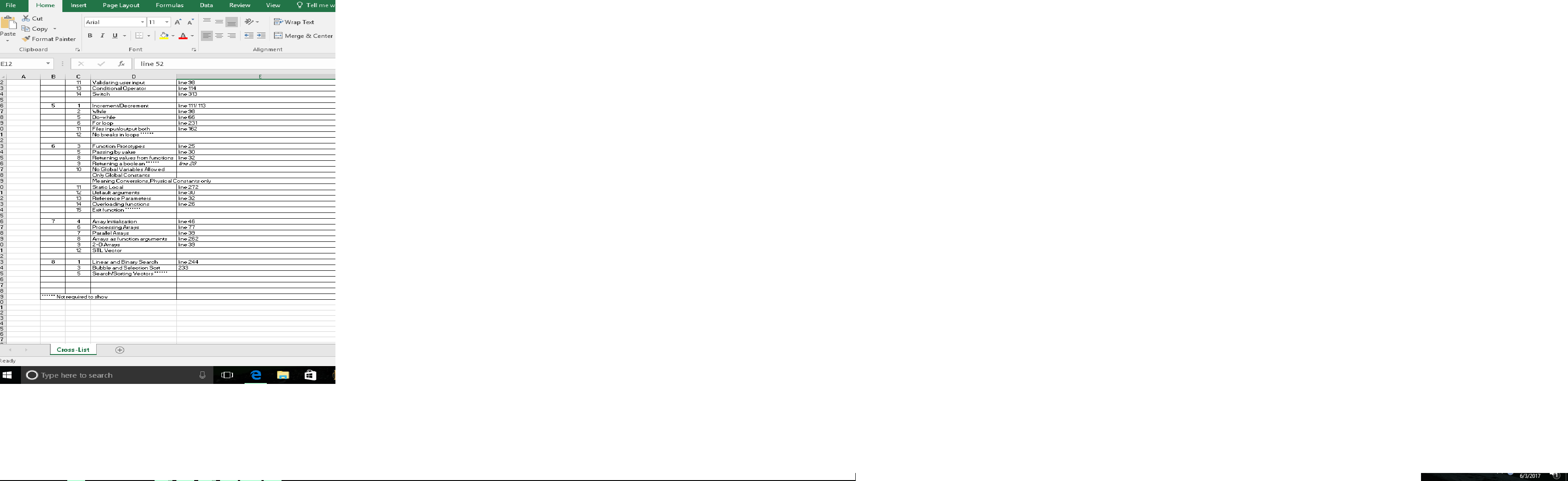
However there are some differences. This game is a 1 player game. To set the ships randomly, I use rand() function to randomize number from 1-6 for x and y axis. The 2D array is set with a value of 0. If at array[i] = 0, we will now change it to array[i] = 1 which indicates that there’s a ship in that array location. If player 1 were to pick that coordinate, array[i] = 2 which indicates that the ship has been sunk. The game ends when all 10 ships randomly placed have been sunk or if player 1 chooses to end the game after every other 2 guesses. With each guesses, the number of hits and misses are then counted up to be used for statistics.

Flowchart

A better, more readable version of the flowchart will be sent to your email.

**Checklist**



References: Dr. Lehr, Savitch 9thEdition, Stackoverflow, Friends of mine who are comfortable with C++

**Code**

/\*

\* File: main.cpp

\* Author: William Gunadi

\* Created on March 13th, 2017,

\* Purpose: Simulate a battleship game.

\*/

//System Libraries

#include <iostream> //Input - Output Library

#include <ctime> //to randomize ship locations

#include <string>

#include <iomanip>

#include <fstream>

#include <cstdlib>

#include <cmath>

using namespace std; //Name-space under which system libraries exist

//User Libraries

//Constants for 2D array.

const int column = 7;

//Function Prototypes

void intro();

void restart(int[][column], int);

void myBoard(int[][column], int);

bool guess(int[][column], int x, int y);

void insrtion();

void perfmnce(int, int, int);

void opFile(ofstream&, int, int, int);

int search(int\*, int, int);

void sortScr(int\*, int);

//Execution begins here

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

//Declare variables

int row = 7, size = 8;

int board[row][column]; //Declare array board

string outScore = "score.txt";

ofstream outData; //File output performance

ofstream outScr; //File output n ships sunk

ifstream inScr; //File input n ships sunk.

outScr.open(outScore, fstream::app); //open outdata file

inScr.open(outScore); //open outScore file

int readScr[] = {1,2,1,0,1,2,1,10};

int maxShip = 10, shpCtr; //Max ship/Ship counter

int shpDown, ckScore;

int num1, num2;

int counter = 0, hits = 0, misses = 0, guesses = 0; //variables for stats

srand(time(0)); //Randomize location of ships

char ans, stop, funny, plyAgn;

float perHit, perMiss;

string name;

cout << fixed << setprecision(2) << showpoint; //Decimals 2 places

//Input data

cout << "Enter name:";

getline(cin,name);

cout << "Hello " << name << "!" << endl;

cout << "This program simulates a battleship game!" << endl;

insrtion();

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

do{

restart(board, row);

myBoard(board, row);

shpCtr = 10;

shpDown = 0;

while (shpDown < maxShip) //Set the ship's location

{

int x = rand() % row; // Randomly place ship in rows

int y = rand() % column; // Randomly place ship in columns

if (board[x][y] != 1) //1 indicates there's a ship in the location

{

shpDown++; //Increase ship until reaches maxShip

board[x][y] = 1;

}

}

//Initiate loop

while(shpCtr != 0) //Play again or not

{

cout << "Please enter two numbers [0-6] to guess location "

<< "of ships" << endl;

cin >> num1; //[0-6] # rows downward

//Input validation for x axis

while (num1 < 0 || num1 > 6)

{

cout << "Number entered is not valid!" << endl;

cout << "Enter a integer between 0-6." << endl;

cin >> num1;

}

cin >> num2;

//Input validation for y axis

while(num2 < 0 || num2 > 6)

{

cout << "Number entered is not a valid number!" << endl;

cout << "Enter an integer between 0-6." << endl;

cin >> num2;

}

//If statement for hits

if(guess(board, num1, num2))

{

cout << "Hit!" << endl;

shpCtr--;

hits++; //counter for # of hits

guesses++; //counter for # of guesses

cout << "number of ships left " << shpCtr << endl;

if ( shpCtr == 0)

{

cout << "Congratulation!!!\n";

cout<< "you've won! game is now ending" << endl;

}

}

//If-statement for misses

else

{

cout << "Miss! Try again!" << endl;

misses++; //counter for # of misses

guesses++;

cout << "Number of ships left: " << shpCtr << endl;

if (guesses % 2 == 0)

{

cout << "Stop playing? If yes, press 'Y'; 'n' to keep going" << endl;

cin >> stop;

if(stop == 'y' || stop == 'Y')

{

shpCtr = 0;

}

}

}

}

//Output the transformed data

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

myBoard(board, row); //Show board

//Output stats

perfmnce(hits, misses, guesses);

cout << endl << "do you like this game? (y/n) ";

cin >> funny;

(funny == 'y')? cout << "awesome," << endl :

cout << "that's too bad" << endl;

if(funny == 'y' || funny == 'Y'){

cout << endl << "since you like this game, would you like to play again?" << endl;

cin >> plyAgn;

}

else

{

cout << endl << "Would you like to play again? (y/n) ";

cin >> plyAgn;

}

}

while (plyAgn == 'y'); //play again

//Output to a file

opFile(outData, hits, misses, guesses);

//output hits to a file

cout << "outputting hits to " << outScore << " file.. " << endl;

outScr << hits;

outScr << endl;

//Input previous scores into an array

if(inScr.fail())

{

cout << "Input file opening failed! " << endl;

exit(1);

}

//Automatically read input file and put into inScr array

/\* NOT WORKING YET

indx = 0;

\* inScr >> readScr[indx];

while(inScr >> readScr[indx])

{

inScr >> readScr[indx];

cout << readScr[indx];

indx++;

}

\*/

//Linear search function..

cout << endl << endl;

cout << "Would you like to search previous player's score?\n"

<< " if so, press 'y'." << endl;

cin >> ans;

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

{

cout << "What score would you like to check? " << endl;

cin >> ckScore;

int temp = search(readScr, 20, ckScore);

if(temp != -1)

{

cout << "Yes, we've found a previous player that scored " << ckScore

<< " at position " << temp+1 << " of the array. " << endl;

}

else

cout << "Nope, no one has sunked " << ckScore << " ships yet." << endl;

cout << "would you like to check a different score?" << endl;

cin >> ans;

}

//Sort the array

cout << endl << endl << "We will attempt sort the scores from lowest to highest"

<< "\nusing a bubble sort." << endl;

sortScr(readScr, size);

for(int i = 1; i <= 8; i++)

{

if(i % 2 == 0)

cout << endl;

cout << readScr[i - 1] << " ";

}

//Close files.

outScr.close();

outData.close();

inScr.close();

return 0;

}

void sortScr(int\* readScr, int size)

{

//Declare local variables

bool swap;

int temp;

do{

swap = false;

for(int i = 0; i < (size - 1); i++)

{

if (readScr[i] > readScr[i + 1])

{

temp = readScr[i];

readScr[i] = readScr[i + 1];

readScr[i + 1] = temp;

swap = true;

}

}

} while (swap);

}

int search(int \*readScr, int size, int ckScore)

{

//declare local variables

int index = 0;

int position = -1;

bool found = false;

//Search algorithm

while(index <= size && !found)

{

if(readScr[index] == ckScore)

{

found = true;

position = index;

}

index++;

}

return position;

}

void restart(int board[][column], int row){

for (int a = 0; a < row; a++)

{

for( int b = 0; b < column; b++)

board[a][b] = 0;

}

}

void myBoard(int board[][column],int row)

{ //Function to display board.

int counter = 0;

for (int a = 0; a < row; a++)

{

for( int b = 0; b < column; b++)

{

cout << "[" << board[a][b] << "]";

}

cout << endl; //Rows

}

}

bool guess(int board[][column], int x, int y) // x's row y's column

{

if(board[x][y] == 1)

{

board[x][y] = 2; //Ship sunk

return true;

}

return false;

}

void insrtion()

{

cout << "There are 10 ships randomly allocated throughout the board" << endl;

cout << "The board has 7 rows and 7 columns starting from 0-6." << endl;

cout << "The game ends when you've correctly chosen all 10\n"

<< "spots where the ships are located or if you'd like to quit." << endl;

}

void perfmnce(int hits, int misses, int guesses)

{

cout << "Time to grade your performance.." << endl;

cout << endl << "number of guesses: " << guesses << endl;

cout << "number of hits: " << hits << endl;

cout << "number of misses: " << misses << endl;

cout << "Percentage of hits: "

<< static\_cast<float>(hits)/guesses\*100 << "%" << endl;

cout << "Percentage of misses: "

<< static\_cast<float>(misses)/guesses\*100 << "%" << endl;

switch(hits)

{

case 10: cout << "You've hit all of the ships! you're good!" << endl; break;

case 9:

case 8:

case 7: cout << "you were almost there..." << endl; break;

case 6: cout << "Could do better.." << endl; break;

case 5:

case 4:

case 3: cout << "You've sunk only 50% or less of the ships. Not good."

<< endl; break;

case 2:

case 1:

case 0: cout << "You need to work on you guessing game " << endl; break;

}

}

void opFile(ofstream& outData ,int hits, int misses, int guesses)

{

cout << "We'll now output your stats into a file.. " << endl;

outData.open("stats.txt, app");

if(outData.fail())

{

cout << "Output file opening failed! " << endl;

exit(1);

}

outData << "Time to grade your performance.." << endl;

outData << endl << "number of guesses: " << guesses << endl;

outData << "number of hits: " << hits << endl;

outData << "number of misses: " << misses << endl;

outData << "Percentage of hits: "

<< static\_cast<float>(hits)/guesses\*100 << "%" << endl;

outData << "Percentage of misses: "

<< static\_cast<float>(misses)/guesses\*100 << "%" << endl;

}