Battleship

Board Game

Danielle Fernandez

CIS 5 - 43518

Spring 2022

Project 2

http://github.com/koa2019/cis5/tree/main/project\_2

TABLE OF CONTENTS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2

INTRODUCTION\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

HOW THE GAME WORKS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

OBJECT OF THE GAME 3

RULES OF THE GAME 3

HOW TO PLAY 3

MY APPROACH TO THE GAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

TRANSLATING GAMEPLAY TO RULES TO PROGRAMMING LANGUAGE 4

SIMILARITIES TO THE BOARD GAME 4

DIFFERENCES FROM THE GAME 4

THE LOGIC OF IT ALL\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5

PSEUDO 5

FLOWCHART 9

CONSTRUCTS & CONCEPTS UTILIZED\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

PROOF OF A WORKING GAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_18

REFERENCES\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20

PROGRAM\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20

**Introduction**

Battleship is a guessing game that dates back to the 1930s around World War I and started as a pencil and paper game. It wasn’t until 1967 that Milton Bradley started to produce it as a plastic board game with pegs. It was one of the earliest board games to be turned into a computer game. It’s even found its way to gaming consoles like Playstation 2, Nintendo Wii, and Xbox. The time-honored classic is still being produced and can be found as a plastic game unit, a card game, or an outer space version.

**How the Board Game Works**

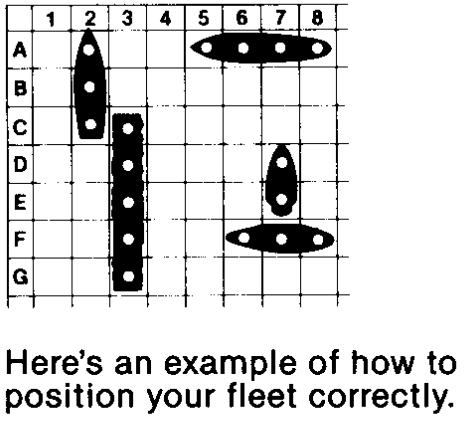
**Object of the Game**

Be the first to sink all 5 of your opponent’s ships.

**Prepare for Battle**

Battleship is a 2 player game that uses 2 game units. You and your opponent sit facing each other, with the lids of your game units raised so neither of you can see the other’s grid.

Secretly place your fleet of 5 ships on your ocean grid. Your opponent does the same.

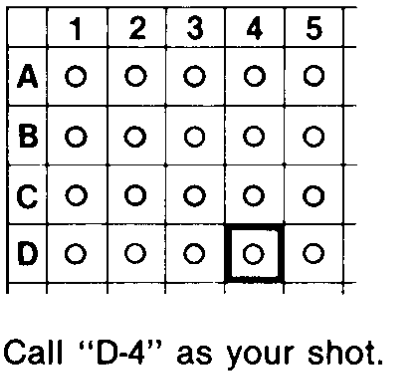


**Rules for placing ships:**

1. Place each ship in any horizontal or vertical position, but not diagonally.
2. Do not place a ship so that any part of it overlaps letters, numbers, the edge of the grid, or another ship.
3. Do not change the position of any ship once the game has begun.

**How to Battle**

Decide who will go first. You and your opponent alternate turns, calling out one shot per turn to try and hit each other’s ships.



On your turn, pick a target hole on your grid and call out its location by letter and number. Each target holds has a letter-number coordinate that corresponds with the same coordinate on your opponent’s ocean grid, To determine each coordinate, find its corresponding letter on the left side of the grid and its number on the top of the grid.

When you call your shot, your opponent must tell whether your shot is a hit or miss.

**Winning the Game**

If you’re the first player to sink your opponent’s entire fleet of 5 ships, you win the game!

**My Approach to the Game**

**Translating Game Play to Rules to Programming Language**

Initial questions I faced when trying to figure out how to convert this game from single data types to arrays and/or 2D arrays:

* “How do I convert the board from single variables to 2D arrays?”
* “Should I also convert the player’s guesses from single randomly generated values to user input?”
* “Should players’ guesses be generated with a random value each time or should I a 2D array using a single array coupled with a random function?”
* “Is there a way to mimic the different number of peg holes each ship has?”

**Problems and Solutions I incurred while coding:**

* Problem: mixing up which of the four 2D arrays I was comparing or displaying.
* Solution: created a print array function & called it in strategic locations
* Problem: requiring both players to have at least 3 ships in each of their randomly generated 2D arrays
* Solution: Required at least 1 player to have a minimum of 3 ships
* Problem: How and when to end the game. Max games and total games controlled version 1, and I wanted the game to play through the entire 2D arrays.
* Solution: Set max games to equal whichever player had the most hits which meant I had to compare the four 2D arrays at the beginning of the game
* Problem: Resetting the game to allow a player to play again
* Solution: Removed “Play again” feature

**Similarities to the Board Game**

My version of the game and the actual game are similar in a few ways. Both games require 2 players. They both utilize a grid layout with letters and numbers for the game board. Players alternate taking guesses. The winner is determined by the first player to sink all of their opponent’s ships.

**Differences from the Board Game**

The games differ in that my version doesn’t give each player a fleet of 5 vessels. My game requires at least one player to have at least 3 ships in their fleet. My board is an 8x2 grid whereas the original uses an 8x7 grid. I automatically generate each player’s guess.

**The Logic of it All**

**Pseudo**

I’ve included the pseudo for my game. My program had over fifteen functions, so I decided to only pseudo functions that illustrated the main ideas we covered in the second half of the semester: passing single arrays and 2D arrays through functions, bubble sort, and selection sort using vectors. I’ve also included a zip folder of project 1 in this project’s folder, so you could reference my progression.

*Include system libraries*

*Declare function prototypes*

*Start main function*

*Declare constants and variables*

*Open file1 that contains player 1’s ship locations*

*Open file2 that contains player 2’s ship locations*

*Create a new file to write to*

*Initialize 4 different counter variables to zero.*

*numShp1=numShp2=count1=count2=0;*

*Start double for loop to run from 1 < 9.*

*Read in file1 and save to character 2D array board1[row][col]*

*If conditional that checks if it equals ‘S’, then increase numShp1 by 1;*

*Increase count1 by 1;*

*Read in file2 and save to character 2D array board2[row][col]*

*If conditional that checks if it equals ‘S’, then increase numShp2 by 1;*

*Increase count2 by 1;*

*Call function with a reference variable to ask for player 1’s name*

*Call function to randonly fill p1Guess[] [] and p2Guess[][] with choices[ ]*

*Call function to show switch menu ( )*

*Call function with a pass-by-value to display the game’s introduction message.*

*Call string function to return player 2's name that is picked randomly from an array*

*Display player 1 and player 2 names*

*Call function to pause game to let the user see who they’re playing*

*Set flags for if p1 guessed correctly and p2 guessed correctly*

*Set row and col indices that control the flow of p1Guess[][], board2[][], p2Guess[][], board1[] [] to 1*

*while loop. Runs until a player correctly guesses their opponent's ship location*

*Call function to display round number banner*

***// PLAYER 1’S TURN BEGINS HERE***

*Call function to display instructions and player 1’s name*

*Set player1Guess to first indices guessP1[row][col]*

*Set player2Ship to first indices board2[row][col]*

*Call function to show player 1’s guess as a letter and number*

*// Compare two 2D arrays to check if player 1’s guess is correct*

*if p1Guess equals ‘S’ & p2Ship equals ‘S’, then*

*decrease player2 number of ships by 1*

*Increase player 1’s number of wins by 1*

*Set isHit to true*

*Call function to display HIT message, p1Guess and p2Ship values*

*Call function to ask if they want to verify it was a hit*

*Call function to validate user's answer. Function returns bool value.*

*if(true), then*

*Call function to display opposing 2D arrays to confirm it was a hit*

*Call pause function to allow user time to view the arrays*

*if (player 1's number of wins equals max games), then*

*Set player 1 correct flag variable to true;*

*else condition for when player 1 is wrong*

*Set isHit to false*

*Call function to display MISS message, p1Guess, p2Ship values*

***// PLAYER 2’S TURN BEGINS HERE***

*if conditional only runs if player 1 was wrong, then*

*Call function to display instructions and player 2’s name*

*Set player2Guess to first indices guessP2[row][col]*

*Set player1Ship to first indices board1[row][col]*

*Call function to show player 2’s guess as a letter and number*

*// Compare two 2D arrays to check if player 1’s guess is correct*

*if p2Guess equals ‘S’ & p1ship equals ‘S’, then*

*decrease player1 number of ships by 1*

*Increase player 2’s number of wins by 1*

*Set isHit to true*

*Call function to display HIT message, p2Guess and p1Ship values*

*Call function to ask if they want to verify it was a hit*

*Call function to validate user's answer. Function returns bool value*

*if(true), then*

*Call function to display opposing 2D arrays to confirm it was a hit*

*Call pause function to allow user time to view the arrays*

*if (player 2's number of wins equals max games), then*

*Set player 2 correct flag to true;*

*else condition for when player 2 is wrong*

*Set isHit to false*

*Call function to display MISS message, p2Guess, p1Ship values*

*// player 2’s turn ends here*

*if col index is less than or equal to 8, then*

*Increment col index by 1*

*if p1Correct is false & p2Correct is false, then*

*if col Index is greater than 8 & row Index is than or equal to 2, then*

*Increment row Index by 1.*

*Reassign col Index to 1*

*// make sure game ends if neither player has guessed their opponent's ship's location*

*correctly*

*if round is equal to or greater than 16, then*

*Set p1Correct and p2Correct flags to true*

*else display try again message when both players guesses are wrong*

*// end of while( )*

*Sum total number of games won & total number rounds played*

*Calculate average number of rounds it took to win*

*Call function to display both player's scores*

*Call pause function to allow user time to view the scores*

***// START SORT & SEARCH SECTION***

*Call main sort & search function*

*Write each player’s wins and number of games played to outFile*

*Close all files being read in and written to*

*return 0*

*// FUNCTION DEFINTIONS PSEUDO*

*void fllGArr(char guessP1[ ][9],char guessP2[ ][9],char choices[ ],int ROWS,int COLS,int &p1GShps, int &p2GShps){*

*Set size for array*

*Set number of ships p1 & p2 have to zero*

*Declare bool.*

*do*

*Set bool to false*

*for loop from 1 < 3*

*for loop from 1 < 9*

*Initialize p1Guess[ ][ ] to a random char from choices[ ]*

*if p1Guess==’S’, then increment p1 ships*

*if they have 3 ships, then set bool to true*

*Repeat to initialie p2Guess[ ][ ]*

*while(bool is false)*

*int binarySrch(string [ ], string &,int){*

*Set bool to false*

*Declare & initialize various index values to their starting positions in array*

*Call function to change string to uppercase letters*

*While bool is false & searching between first and last indices*

*If array[middle] equals string, then*

*Set bool to true*

*Set position to middle index*

*else if array[middle] is bigger than string, then*

*last equals middle index minus 1*

*Else first equals middle index plus 1*

*Return position*

*bool isReady(char ans){*

*if..if else…else to check ans equals a ‘y’ or ‘Y’*

*void sortBub(string names[ ], int size){*

*Declare bool*

*Declare & initialize int to last index in an array*

*do*

*Set bool to false*

*Loop through array*

*Compare neighboring indices against each other*

*if index on left is greater than its neighbor, then*

*swap their positions in array*

*Reassign bool to true*

*Drecrement int by 1*

*while(bool is true)*

*Sort names in alphabetically order*

*void sortSel(vector<string> &){*

*lor range loop through vector and print each name*

*Loop through vector 0 through size-1*

*Set min to zero*

*Set smallest value to first index in vector*

*Loop trough vector starting at 1 to vector.size()*

*If vector[index] is less than smallest value, then*

*Smallest value equals that vector[index]*

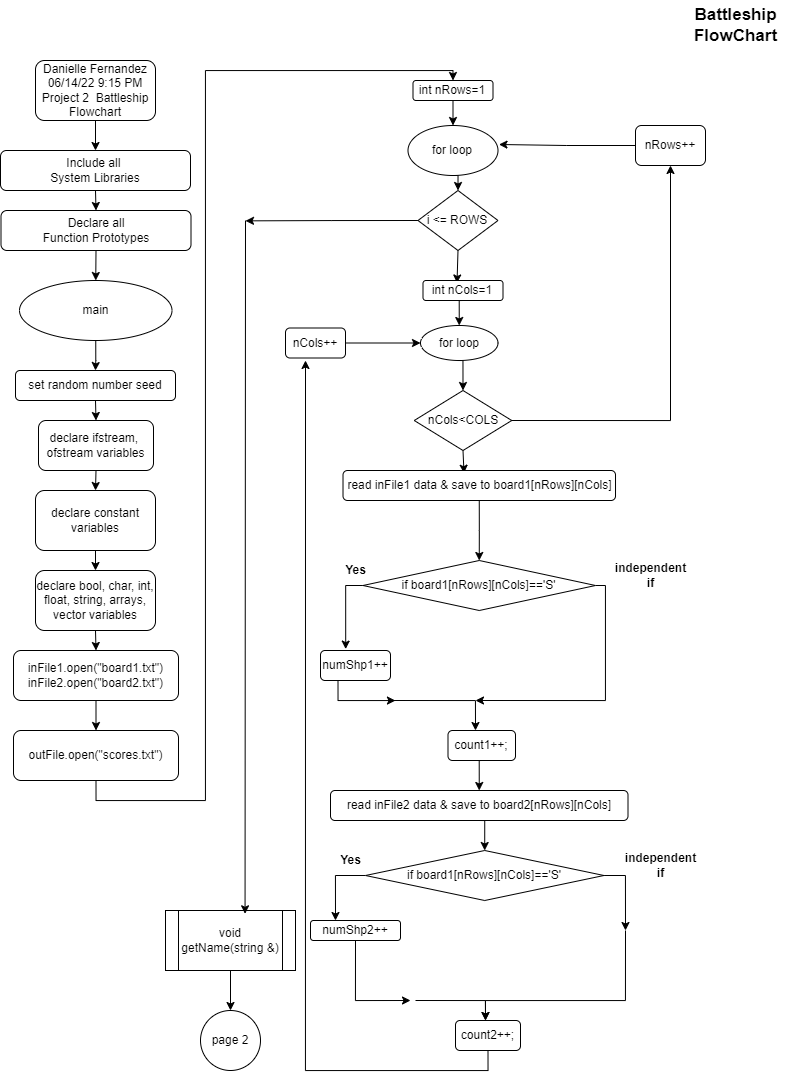
*Reassign min to current index value*

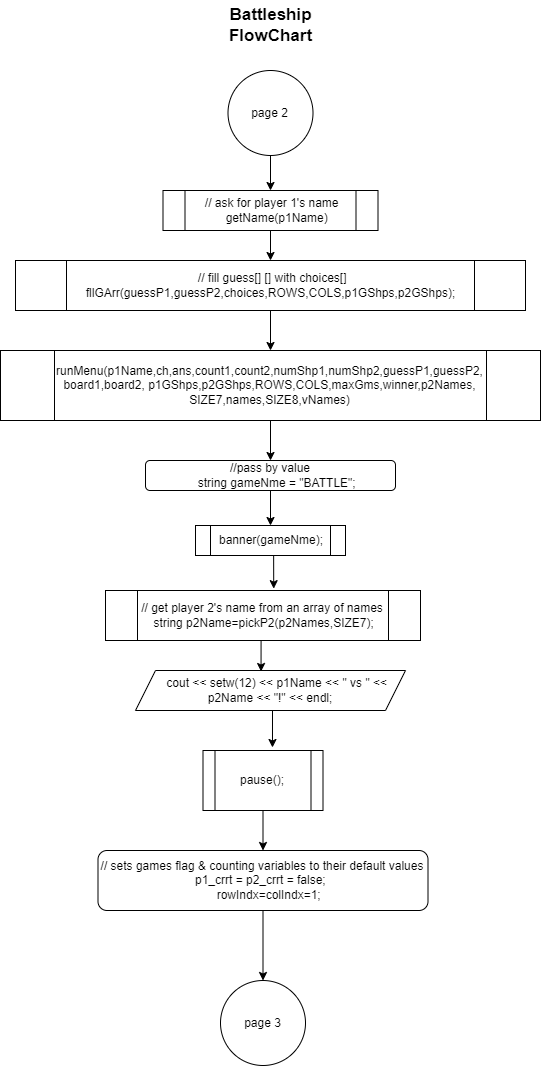
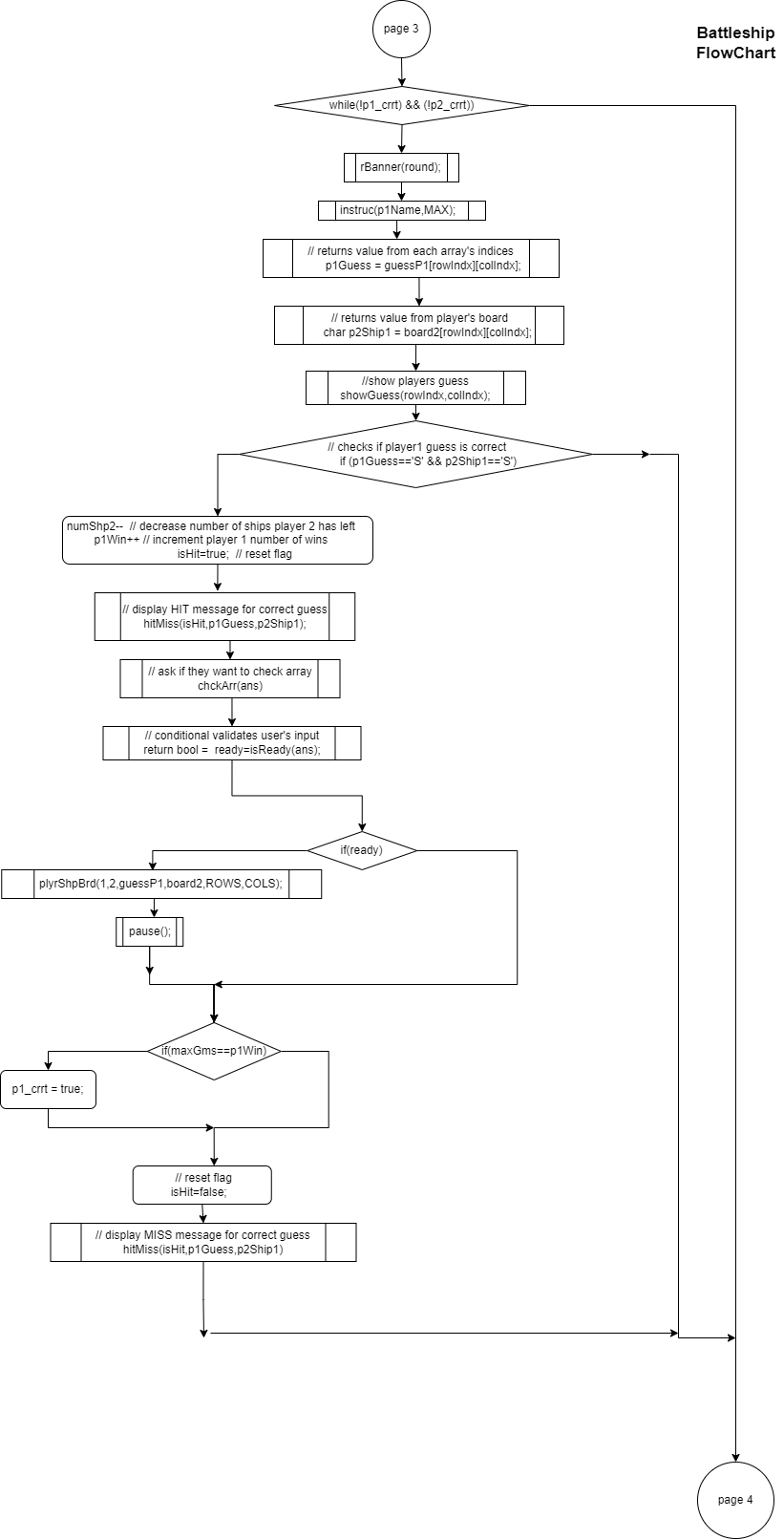
*Swap their positions in vector*

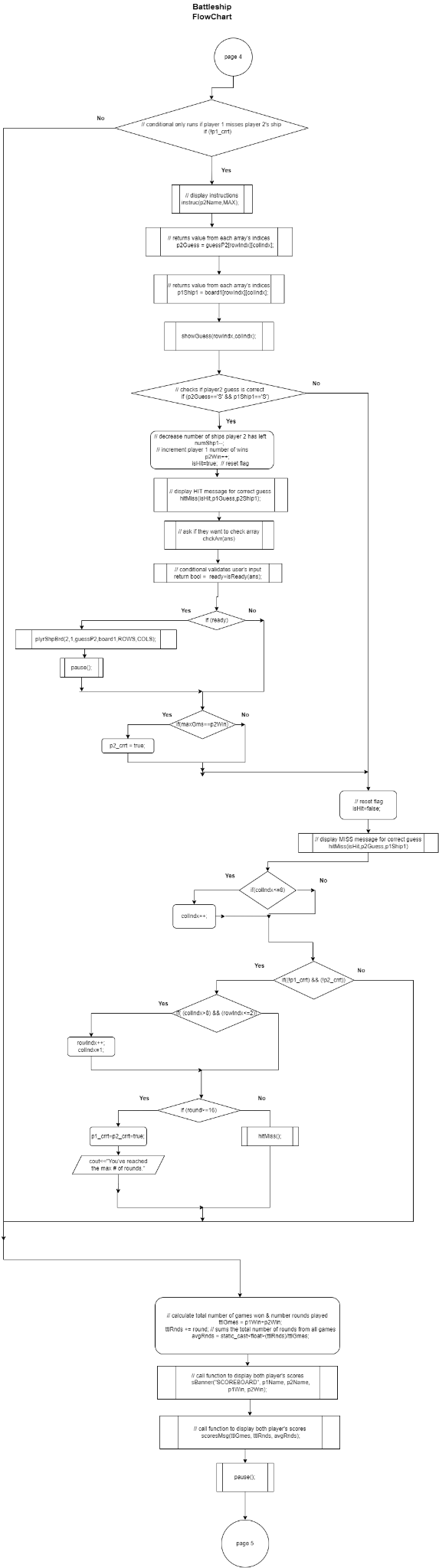
*Print vector of names in alphabetical order*

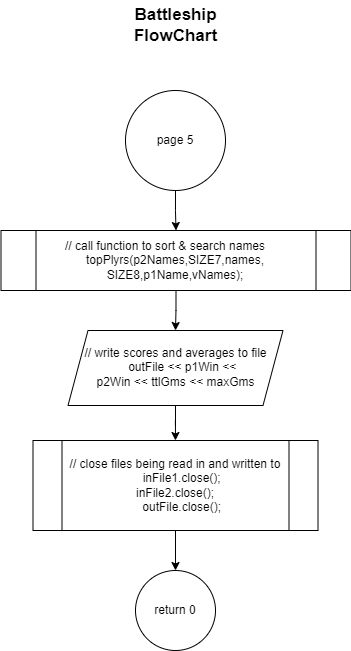
**Flowchart**

My flowchart is very long, so I have included pictures of the main function. You can find the original in the documents folder within project 2.









**Constructs & Concepts Utilized**

A copy of this table is located in the documents folder within project 2.

| 2 | 2 | cout |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 3 | libraries | 10-18 | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | variables/literals |  |  | No variables in global area, failed project! |
|  | 5 | Identifiers |  |  |  |
|  | 6 | Integers | 93 | 1 |  |
|  | 7 | Characters | 86 | 1 |  |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
|  | 8 | Strings | 170 | 1 |  |
|  | 9 | Floats No Doubles | 109 | 1 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 81 | 1 |  |
|  | 11 | Sizeof \*\*\*\*\* | \*\*\*\*\* |  |  |
|  | 12 | Variables 7 characters or less | X |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables | X |  |  |
|  | 14 | Arithmetic operators | 292 |  |  |
|  | 15 | Comments 20%+ | X | 2 | Model as pseudo code |
|  | 16 | Named Constants | 73 |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate | X |  | Emulate style in book/in class repositiory |
|  |  |  |  |  |  |
| 3 | 1 | cin |  |  |  |
|  | 2 | Math Expression |  |  |  |

| 2 | 2 | cout |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 3 | libraries | 10-18 | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | variables/literals |  |  | No variables in global area, failed project! |
|  | 5 | Identifiers |  |  |  |
|  | 6 | Integers | 93 | 1 |  |
|  | 7 | Characters | 86 | 1 |  |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
|  | 8 | Strings | 170 | 1 |  |
|  | 9 | Floats No Doubles | 109 | 1 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 81 | 1 |  |
|  | 11 | Sizeof \*\*\*\*\* | \*\*\*\*\* |  |  |
|  | 12 | Variables 7 characters or less | X |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables | X |  |  |
|  | 14 | Arithmetic operators | 292 |  |  |
|  | 15 | Comments 20%+ | X | 2 | Model as pseudo code |
|  | 16 | Named Constants | 73 |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate | X |  | Emulate style in book/in class repositiory |
|  |  |  |  |  |  |
| 3 | 1 | cin |  |  |  |
|  | 2 | Math Expression |  |  |  |
|  | 3 | Mixing data types \*\*\*\* | \*\*\*\*\* |  |  |
|  | 4 | Overflow/Underflow \*\*\*\* | \*\*\*\*\* |  |  |
|  | 5 | Type Casting | 294 | 1 |  |
|  | 6 | Multiple assignment \*\*\*\*\* | \*\*\*\*\* |  |  |
|  | 7 | Formatting output | 175 | 1 |  |
|  | 8 | Strings | 170, 297 | 1 |  |
|  | 9 | Math Library | 456 | 1 | All libraries included have to be used |
|  | 10 | Hand tracing \*\*\*\*\*\* | \*\*\*\*\* |  |  |
|  |  |  |  |  |  |
| 4 | 1 | Relational Operators | 271 |  |  |
|  | 2 | if | 146 | 1 | Independent if |
|  | 4 | If-else | 203-224 | 1 |  |
|  | 5 | Nesting | 203-227 | 1 |  |
|  | 6 | If-else-if | 561-570 | 1 |  |
|  | 7 | Flags \*\*\*\*\* | 184 |  |  |
|  | 8 | Logical operators | 274 | 1 |  |
|  | 11 | Validating user input | 640-645 | 1 |  |
|  | 13 | Conditional Operator | 461 | 1 |  |
|  | 14 | Switch | 628 | 1 |  |
|  |  |  |  |  |  |
| 5 | 1 | Increment/Decrement | 247, 248 | 1 |  |
|  | 2 | While | 188 | 1 |  |
|  | 5 | Do-while | 421-436 | 1 |  |
|  | 6 | For loop | 391 | 1 |  |
|  | 11 | Files input/output both | 145,150,310 | 2 |  |
|  | 12 | No breaks in loops \*\*\*\*\*\* | X |  | Failed Project if included |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  | Total | 30 |  |

\*\*\*\*\*\* Not required to show

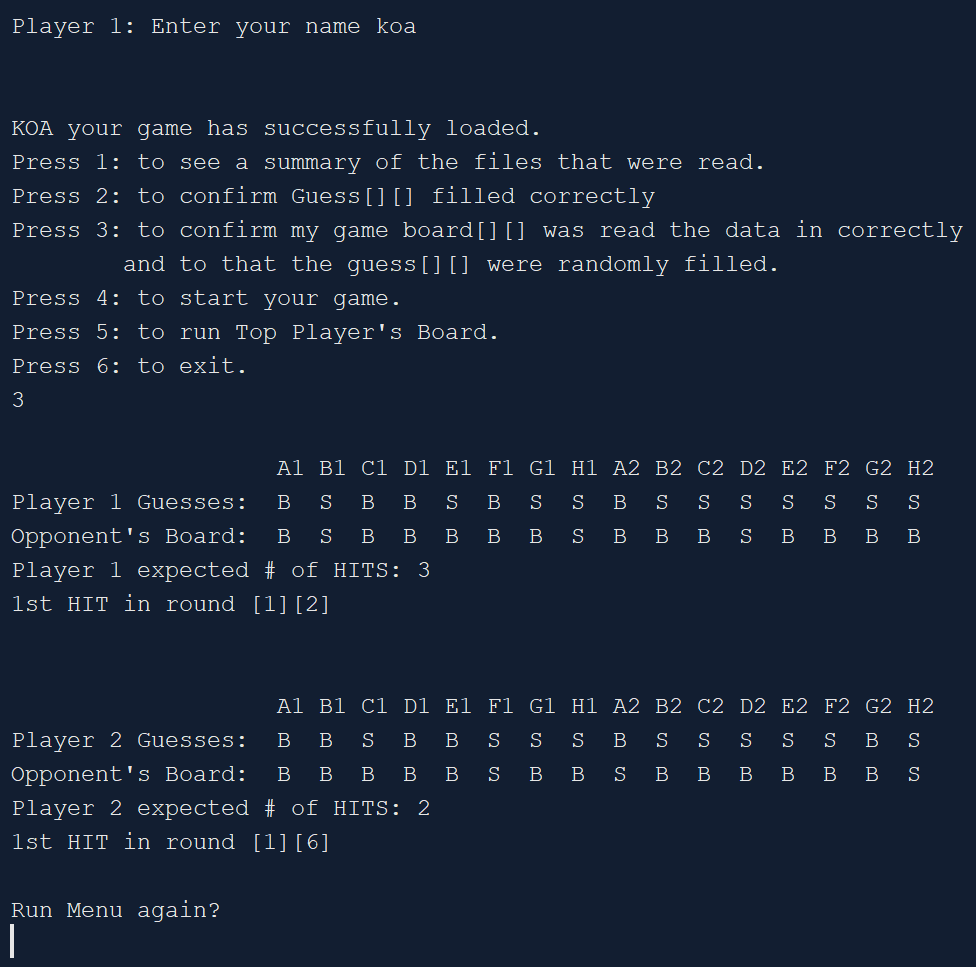
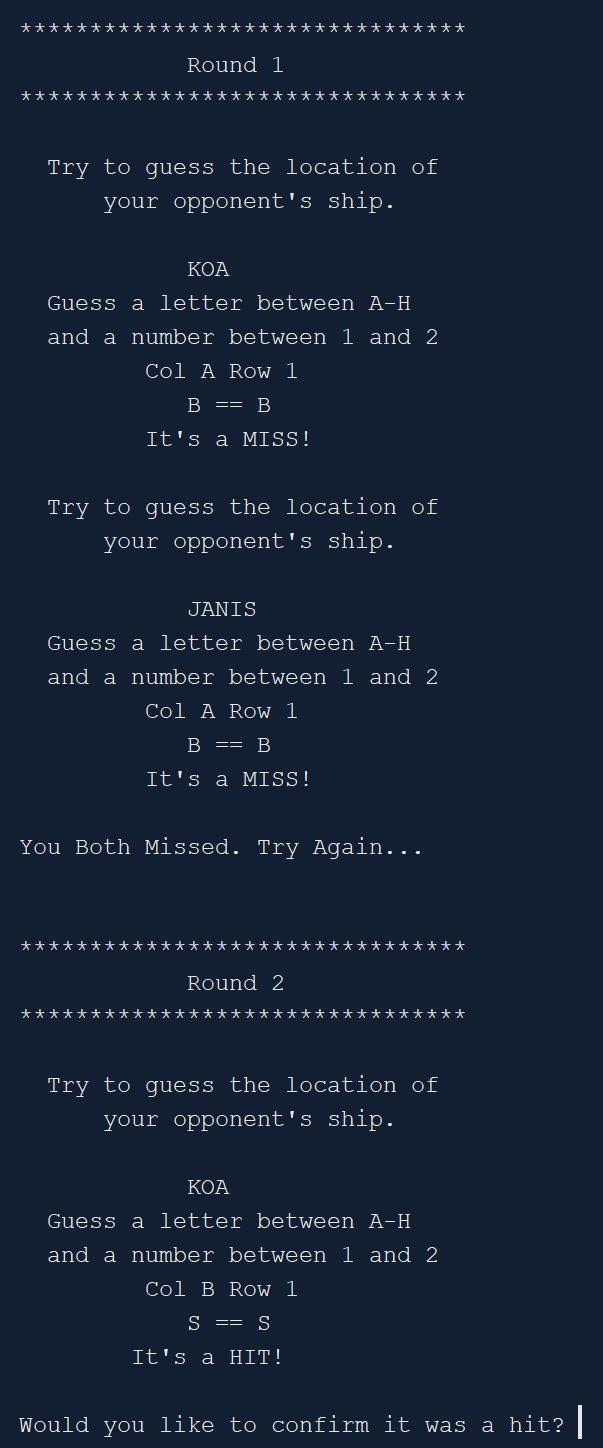
**Cross Reference for Project 2 You are to fill-in with where located in code**

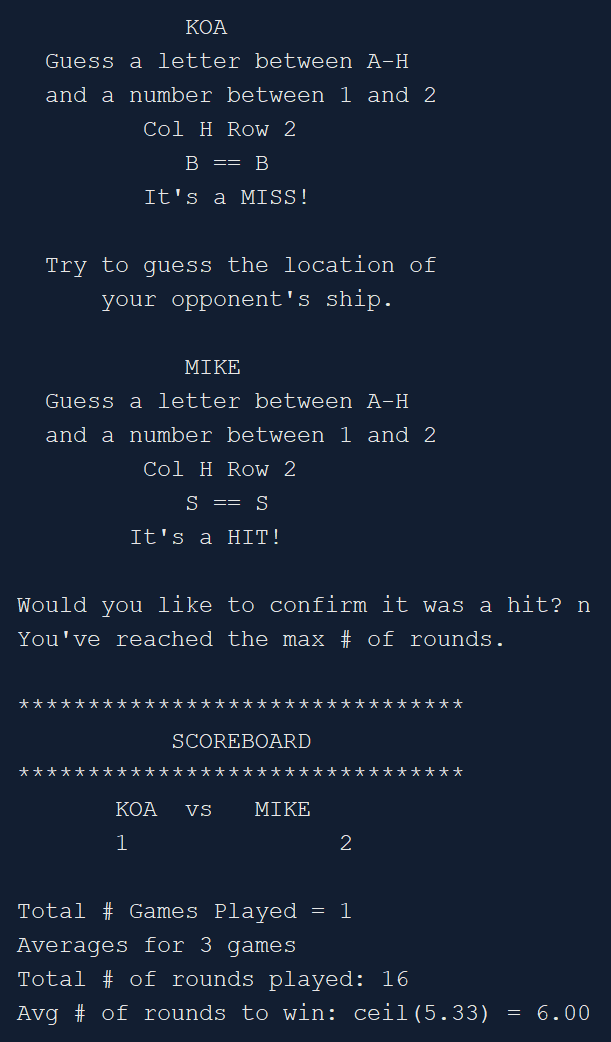
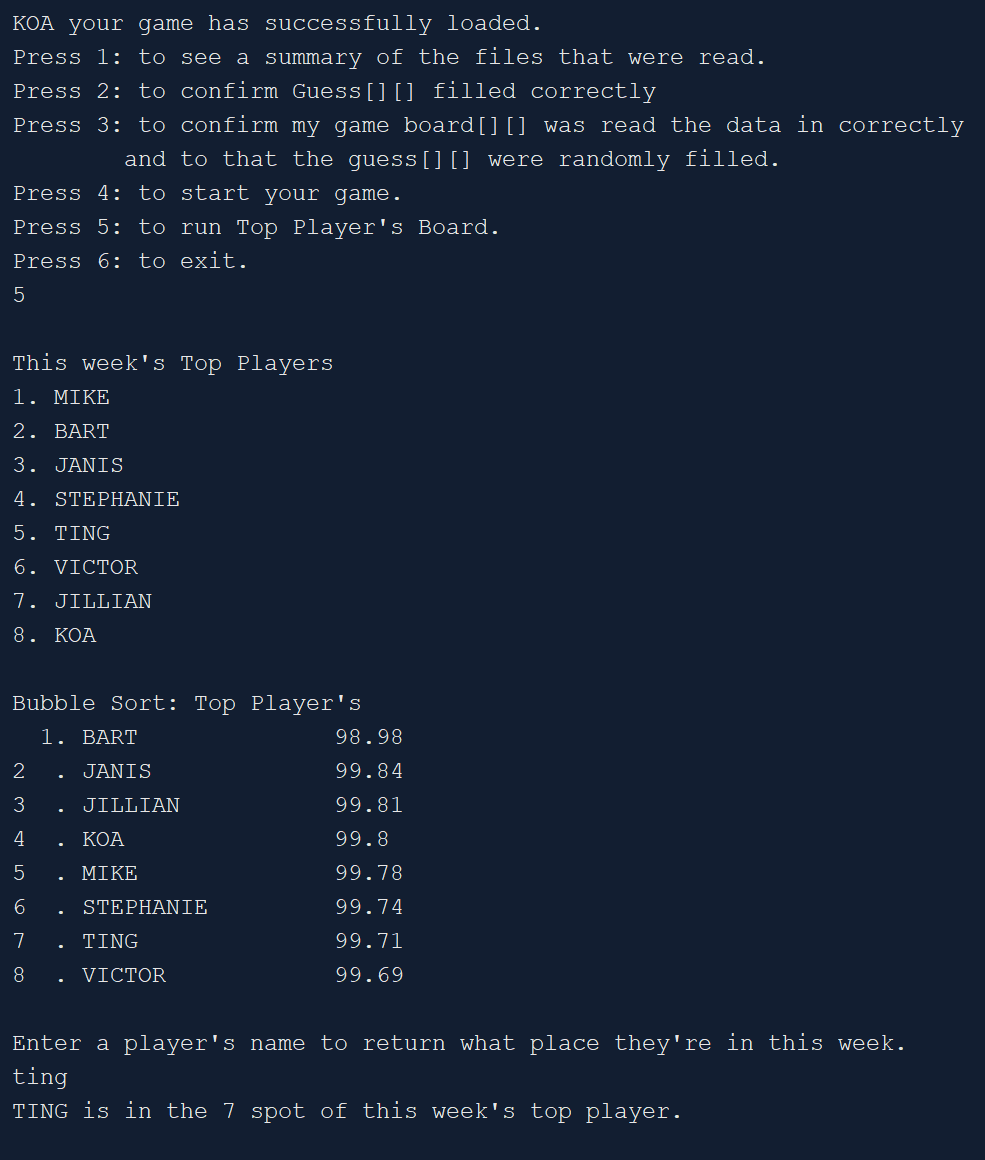
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| --- | --- | --- | --- | --- | --- |
| 6 |  | Functions |  |  |  |
|  | 3 | Function Prototypes | 28-59 | 4 | Always use prototypes |
|  | 5 | Pass by Value | 170 | 4 |  |
|  | 8 | return | 793 | 4 | A value from a function |
|  | 9 | returning boolean | 255, 649 | 4 |  |
|  | 10 | Global Variables | X | XXX | Do not use global variables -100 pts |
|  | 11 | static variables | 451,476 | 4 |  |
|  | 12 | defaulted arguments | 38 | 4 |  |
|  | 13 | pass by reference | 612 | 4 |  |
|  | 14 | overloading | 36, 37, 48, 49 | 5 |  |
|  | 15 | exit() function | 635 | 4 |  |
| 7 |  | Arrays |  |  |  |
|  | 1 to 6 | Single Dimensioned Arrays | 120 | 3 |  |
|  | 7 | Parallel Arrays | 357 | 2 |  |
|  | 8 | Single Dimensioned as Function Arguments | 612, 692 | 2 |  |
|  | 9 | 2 Dimensioned Arrays | 612, 692 | 2 | Emulate style in book/in class repositiory |
|  | 12 | STL Vectors | 123, 342 | 2 |  |
|  |  | Passing Arrays to and from Functions | 612 | 5 |  |
|  |  | Passing Vectors to and from Functions | 375, 379 | 5 |  |
|  |  |  |  |  |  |
| 8 |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort | 416 | 4 |  |
|  | 3 | Selection Sort | 379 | 4 |  |
|  | 1 | Linear or Binary Search | 767 | 4 |  |
|  |  |  | Total | 70 | Other 30 points from Proj 1 first sheet tab |

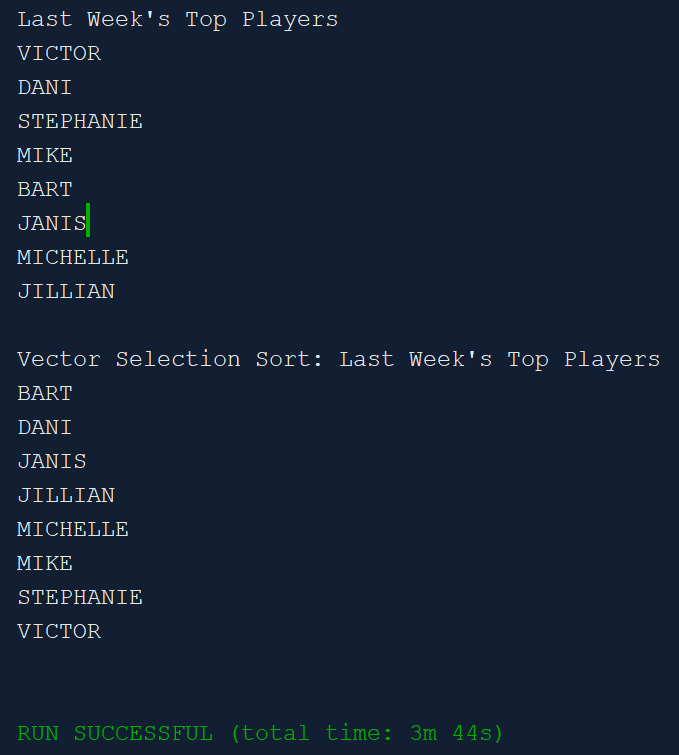
\*\*\*\*\*\* Not required to show

**Proof of a Working Product**

Full images of these screenshots are located in the images folder within project 2 folder.

****

** **

****

**References**

1. Dr. Lehr’s github. https://github.com/ml1150258/2022\_Spring\_CSC\_CIS\_5.
2. Hasbro Battleship Instructions. https://www.hasbro.com/common/instruct/battleship.pdf.

**Program**

/\*

\* File: main.cpp

\* Author: Danielle Fernandez

\* Created on June 4, 2022, 4:28 PM

\* Purpose: Project 2. Covers chapters 1-7 in Gaddis.

\* Version 6: implement vector and pass it to and from functions

\*/

// System Libraries:

#include <iostream> // cin, cout

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

#include <cmath> // round()

#include <cstdlib> // rand(), EXIT\_SUCCESS

#include <fstream> // ofstream

#include <cstring> // string library

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

#include <cctype> // toupper()

#include <vector> // vector

using namespace std;

// User libraries

// Global Constants

// Physics/Chemistry/Math/Conversions

// Function prototypes

void banner(string); // display game

int binarySrch(string [],string &,int);

void cGssArr(const char [][9], const char [][9],const int,const int,int, int); // confirms guess[][] filled correctly

void chckArr(char &);

void copyAdd(string [],const int,string[],const int,string);

void fileSum(int,int,int,int);

void fllGArr(char [][9],char [][9],char [], int, int, int &, int&);

void getName(string &); // get player 1's name using pass by reference

void hitMiss(bool,int,int); // hit message for correct guess

void hitMiss(); // try again message

void instruc(string, const int, int =1); // instructions for players

bool isReady(char); // returns bool value

void runMenu(string, char &, char &, int,int,int,int,char [][9],char [][9],

char [][9],char [][9],int,int,int,int,int &,int&,string [],int,

string [],int,vector<string> &);

void pause();

string pickP2(string [],int); // randomly picks a number for player 2

void plyrShpBrd(int,int,const char [][9],const char [][9],const int,const int);

void print2DArr(const char [][9],const char [][9],

const char [][9],const char [][9],const int,const int,int &,int &);

void prntArr(string []); // prints names[]

void prntArr(const char [][9],int,int); //prints 2D

void rBanner(int &); // display the round number

void sBanner(string,string,string,int,int); // display scoreboard banner

void sortSel(vector<string> &);

void scoresMsg(int,int,float); // displays scores for both players

void showGuess(int,int);

void showStatic();

void sortBub(string [],int); // sort names

void swap1(string &,string &);

void topPlyrs(string [],int,string [],int,string,vector<string> &);

void upper(string &); // changes string to all uppercase letters

// Program execution begins here

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

// set random number seed

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

// declare variables

ifstream inFile1; // for reading an existing file

ifstream inFile2;

ofstream outFile; // for outputting to a file

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

MAX = 16; // maximum number for rand()

const int SIZE7 = 7, // size for player 2 names array

SIZE8 = 8;

const int ROWS = 3,

COLS = 9; // number of cols in 2D array

const int SIZE17=17; // choice array size

bool p1\_crrt, // player 1 correct

p2\_crrt, // player 2 correct

ready, // ready to continue playing

isHit;

char ans, // answer

ch,

p1Guess=0, // player 1 guess

p2Guess, // player 2 guess

p1Ship1, // player 1 ship number 1

p2Ship1; // player 2 ship number 1

int rowIndx,// index for comparing player's guess to their opponent's board

colIndx, // index for comparing player's guess to their opponent's board

maxGms=0, // number of games

round=0, // round

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

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

ttlGmes=0, // sum of both players number of wins

ttlRnds=0, // sum of total rounds played

p1GShps,

p2GShps,

numShp1,

numShp2,

count1,

count2,

winner;

float avgRnds; // average rounds it takes to win

string p1Name = " ",

p2Name = " ";

char board1[ROWS][COLS]={};

char board2[ROWS][COLS]={};

char guessP1[ROWS][COLS]={};

char guessP2[ROWS][COLS]={};

// will be used to fill each player's guess[][]

char choices[SIZE17]={'S','B','S','S','B','S','S','B','S','S','B','S','S','B','S','S','B'};

string p2Names[SIZE7]={"MIKE", "BART", "JANIS", "STEPHANIE", "TING", "VICTOR", "JILLIAN"};

string names[SIZE8]={}; // create new array to hold player 1 & player 2's names

vector<string> vNames{"VICTOR", "DANI", "STEPHANIE", "MIKE", "BART", "JANIS", "MICHELLE", "JILLIAN"};

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

// \*\*\*\*\*\* SET UP GAME STARTS HERE \*\*\*\*\*\*\*\*

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

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

//inFile.open("maxNGms.txt");

inFile1.open("board1.txt",ios::in);

inFile2.open("board2.txt",ios::in);

outFile.open("scores.txt"); // create a file to output to

// initialize counters to zero

numShp1=numShp2=count1=count2=0;

// read in data from file and initialize each player's game board[][]

// with a S(ship) or a B(blank)

for(int nRows=1; nRows<ROWS; nRows++){

for(int nCols=1; nCols<COLS ;nCols++){

inFile1 >> board1[nRows][nCols];

if(board1[nRows][nCols]=='S') {

numShp1++; // count how many ships player 1 has in their array

}

count1++; // count how many items were read in

inFile2 >> board2[nRows][nCols];

if(board2[nRows][nCols]=='S'){

numShp2++; // count how many ships player 1 has in their array

}

count2++; // count how many items were read in

}

}

// ask for player 1's name

getName(p1Name);

// fill guess[] [] with choices[]

fllGArr(guessP1,guessP2,choices,ROWS,COLS,p1GShps,p2GShps);

// show switch menu

runMenu(p1Name,ch,ans,count1,count2,numShp1,numShp2,guessP1,guessP2,board1,board2,

p1GShps,p2GShps,ROWS,COLS,maxGms,winner,p2Names,SIZE7,names,SIZE8,vNames);

// display game's introduction message

string gameNme = "BATTLE";

banner(gameNme);

// get player 2's name from an array of names

p2Name = pickP2(p2Names,SIZE7);

cout << setw(12) << p1Name << " vs " << p2Name << "!" << endl;

pause();

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

//\*\*\*\*\*\*\*\* GAME STARTS HERE\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

// sets games flag & counting variables to their default values

p1\_crrt = p2\_crrt = false;

rowIndx=colIndx=1;

// loops until a player correctly guesses opponents ship location

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

rBanner(round); // display round number banner

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

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

instruc(p1Name,MAX); // display instructions to player 1

// set variables to the their 1st array's indices

p1Guess = guessP1[rowIndx][colIndx];

p2Ship1 = board2[rowIndx][colIndx];

showGuess(rowIndx,colIndx); // show players guess

// checks if player1 guess is correct

if((p1Guess=='S') && (p2Ship1=='S')){

numShp2--; // decrease number of ships player 2 has left

p1Win++; // increment player 1 number of wins

isHit=true; // reset flag

hitMiss(isHit,p1Guess,p2Ship1); // display HIT message for correct guess

chckArr(ans); // ask if they want to check arrays

// validates user's input

ready=isReady(ans);

if(ready) {

// display opposing 2D arrays to confirm it was a hit

plyrShpBrd(1,2,guessP1,board2,ROWS,COLS);

pause();

}

// checks if player 1's number of wins equals max games

if(maxGms==p1Win) p1\_crrt = true; // reassign player 1's value for a correct guess

} else { // else when player 1 guess is wrong

isHit=false; // reset flag

hitMiss(isHit,p1Guess,p2Ship1); // display MISS message for wrong guess

}

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

if(!p1\_crrt){

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

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

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

instruc(p2Name,MAX); // display instructions to player 2

// program generates random number guess

p2Guess = guessP2[rowIndx][colIndx];

p1Ship1= board1[rowIndx][colIndx];

showGuess(rowIndx,colIndx);

// conditional checks if player 2's guess is correct

// program automatically generated guess for player 2

if((p2Guess=='S') && (p1Ship1=='S')){

numShp1--; // decrement number of ships player 1 has

p2Win++; // increment player 2 number of wins

isHit=true; // reset flag

hitMiss(isHit,p2Guess, p1Ship1); // display HIT message for correct guess

chckArr(ans);

// conditional validates user's input

ready=isReady(ans);

if(ready){

plyrShpBrd(2,1,guessP2,board1,ROWS,COLS);

pause();

}

// checks to see if player has won the

if(maxGms==p2Win) p2\_crrt = true; // reassign player 1's value for a correct guess

} else { // else when player 2 guess is wrong

isHit=false;

hitMiss(isHit,p2Guess,p1Ship1); // display MISS message for player 2's wrong guess

}

} // ends player 2's turn

// increment index that controls the guess[][] & board[][]

if(colIndx<=8) colIndx++;

// if both players guessed wrong, then

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

// increases row and resets column

if( (colIndx>8) && (rowIndx<=2)) {

rowIndx++;

colIndx=1;

}

// makes sure game ends if neither player has guessed their opponents ship's location correctly

if(round>=16){

p1\_crrt=p2\_crrt=true;

cout<<"You've reached the max # of rounds.\n\n";

} else hitMiss(); // display try again message when both players guess wrong

}

} // ends while(!p1\_crrt) && (!p2\_crrt)

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

ttlGmes = p1Win+p2Win;

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

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

// call function to display both player's scores

sBanner("SCOREBOARD", p1Name, p2Name, p1Win, p2Win);

scoresMsg(ttlGmes, ttlRnds, avgRnds);

pause(); // pause game so player can see the scores

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

// \*\*\*\*\*\*\*\* SORT & SEARCH NAMES SECTION \*\*\*\*\*\*\*\*\*

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

// call function to sort & search names

topPlyrs(p2Names,SIZE7,names,SIZE8,p1Name,vNames);

// write scores and averages to file

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

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

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

<< ttlGmes << " of " << maxGms << " max games were played.\n";

// close files being read in and written to

inFile1.close();

inFile2.close();

outFile.close();

// exit code

return 0;

}

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTION DEFINITIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

// changes string to all capital letters

void upper(string &name){

char ch;

string temp="";

for(int i=0;i<name.length();i++){

ch = toupper(name[i]);

temp +=ch;

}

name=temp;

}

// function controls the sort and search section

void topPlyrs(string p2Names[],int SIZE7,string names[],int SIZE8, string p1Name,

vector<string> &vNames){

// copy contents of one array to another array and add player1's name

copyAdd(p2Names,SIZE7,names,SIZE8,p1Name);

// print names array

cout << "\nThis week's Top Players \n";

prntArr(names);

// bubble sort names & print sorted array

sortBub(names,SIZE8);

float scores[SIZE8]={98.98, 99.84, 99.81, 99.80, 99.78, 99.74, 99.71, 99.69};

cout << "Bubble Sort: Top Player's \n";

// print parallel arrays

for(int i=0; i<SIZE8; i++){

cout << setw(3)<< i+1 << ". "<<setw(12)

<< left << names[i] << setw(6) << " "

<< scores[i] << endl;

}

cout << endl;

string sName;

cout << "Enter a player's name to see their highest score\n";

cin >> sName;

int score = binarySrch(names,sName,SIZE8);

if(score==-1){

cout << "Sorry, " << sName << " wasn't found.\n";

} else cout << sName << " is in the " << score+1 << " spot of this week's top player.\n\n";

pause(); // pause game so player can view the search findings

// selection sort

sortSel(vNames);

}

// selection sort with vectors

void sortSel(vector<string> &vNames){

cout << "\nLast Week's Top Players \n";

for(auto ele : vNames)

cout << ele << endl;

cout << endl;

int minIndx, last=0;

string minVal;

last=(vNames.size()-1); // don't need to go the last index cause we check it by the end of loop's 1st run

for(int start=0; start<last; start++){

minIndx = start; // set smallest index to zero

minVal=vNames[start]; // set smallest value to array's 1st index

// start loop at array's[1]

for(int indx=(start+1);indx<vNames.size();indx++){

// if its neighbor index is smaller than it

if(vNames[indx] < minVal){

// reassign smallest to that array's index and its value

minVal = vNames[indx];

minIndx = indx;

}

} // swap their values

swap1(vNames[minIndx], vNames[start]);

}

cout << "Vector Selection Sort: Last Week's Top Players \n";

for(auto ele : vNames)

cout << ele << endl;

cout << endl;

}

// bubble sort. compare neighboring indices one at a time

void sortBub(string names[], int size){

bool swap;

int maxElmt=size-1; // (8-1)=7. holds subscript of last element that will be compared to its neighbor

do {

swap=false; // set flag

// loop 0 thru 7

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

if(names[i] > names[i+1]){ // "VICTOR" > "JILLIAN" ?

// swap index values

swap1(names[i],names[(i+1)]);

// reset flag value

swap=true;

}

}

maxElmt--; // decrement. biggest value is at last index, so now restart loop to run to one less last index

} while(swap); // while there's still indices to left to swap

}

// swap indices

void swap1(string &a, string &b){

string temp;

temp=a;

a=b;

b=temp;

}

// displays each players scores and win average

void scoresMsg(int ttlGmes, int ttlRnds, float avgRnds){

showStatic(); // show how many games have been played

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

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

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

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

<< ceil(avgRnds) << endl;

}

// show each players guess

void showGuess(int row,int col){

char colLttr = col==1 ? 'A':

col==2 ? 'B':

col==3 ? 'C':

col==4 ? 'D':

col==5 ? 'E':

col==6 ? 'F':

col==7 ? 'G':

col==8 ? 'H': '?';

cout << setw(13) << "Col " << colLttr

<< " Row " << row <<endl;

}

// static variable for games

void showStatic(){

static int games=1;

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

games++;

}

// display scoreboard banner

void sBanner(string str, string p1Name, string p2Name, int p1Win, int p2Win){

// Display scoreboard banner

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

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

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

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

}

cout << setw(7) << " " << p1Name << setw(4) << "vs" << setw(3) << " "

<< right << p2Name << endl;

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

}

// banner displays round number

void rBanner(int &r){

++r;

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

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

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

}

// prntArr []

void prntArr(const char arr[][9],int ROWS, int COLS){

for(int pRows=1; pRows<ROWS; pRows++){

for(int pCols=1; pCols<COLS; pCols++){

cout << arr[pRows][pCols] << " ";

}

}

}

// prntArr names[]

void prntArr(string arr[]){

for(int i=0;i<8;i++){

cout << i+1 << ". " << arr[i] << endl;

}

cout << endl;

}

//

void print2DArr(const char guessP1[][9], const char guessP2[][9],

const char board1[][9],const char board2[][9],

const int ROWS,const int COLS,int &maxGms,int &winner){

unsigned int hit1,hit2,rIndx1,rIndx2,cIndx1,cIndx2;

char g1,g2,b1,b2;

hit1=hit2=rIndx1=rIndx2=cIndx1=cIndx2=0;

// get how many ship hits each player should get during this game

for(int pRows=1; pRows<ROWS; pRows++){

for(int pCols=1; pCols<COLS; pCols++){

g1=guessP1[pRows][pCols];

b2=board2[pRows][pCols];

if((g1=='S') && (b2=='S')){

hit1++;

if(hit1==1){

rIndx1=pRows;

cIndx1=pCols;

}

}

}

}

for(int row2=1; row2<ROWS; row2++){

for(int col2=1; col2<COLS; col2++){

g2=guessP2[row2][col2];

b1=board1[row2][col2];

if((g2=='S') && (b1=='S')){

hit2++;

if(hit2==1){

rIndx2=row2;

cIndx2=col2;

}

}

}

}

// sets max number of games and returns which player to use for a conditional in main()

if(hit1>hit2){

maxGms=hit1;

winner=1;

}else if(hit1<hit2) {

maxGms=hit2;

winner=2;

} else{

maxGms=hit1;

winner=2;

}

plyrShpBrd(1,2,guessP1,board2,ROWS,COLS);

cout << "\nPlayer 1 expected # of HITS: "<< hit1 << endl

<< "1st HIT in round [" << rIndx1<<"]["<<cIndx1<<"]" <<endl<<endl;

plyrShpBrd(2,1,guessP2,board1,ROWS,COLS);

cout << "\nPlayer 2 expected # of HITS: " << hit2 << endl

<< "1st HIT in round [" << rIndx2 <<"]["<<cIndx2<<"]" <<endl<<endl;

}

//

void plyrShpBrd(int player,int opponnt,const char guess[][9],const char board[][9],

const int ROWS,const int COLS){

cout << endl << setw(19)<<" "<< "A1 B1 C1 D1 E1 F1 G1 H1 A2 B2 C2 D2 E2 F2 G2 H2\n";

cout << "Player "<<player<< " Guesses: ";

prntArr(guess,ROWS,COLS);

cout << endl;

cout << "Opponent's Board: " ;

prntArr(board,ROWS,COLS);

}

// randomly picks a name from an array as player 2's name

string pickP2(string p2Names[], int SIZE7){

cout << "\nLocating your opponent online......\n";

string name2=p2Names[rand()%(SIZE7)];

return name2;

}

// pause screen before game starts

void pause(){

cout << "\nPress enter to continue. ";

cin.ignore();

cin.get();

}

// prompt user

void runMenu(string p1Name, char &ch, char &ans, int count1,int count2,int numShp1,int numShp2,

char guessP1[][9],char guessP2[][9],char board1[][9],char board2[][9],

int p1GShps,int p2GShps,int ROWS,int COLS,int &maxGms, int &winner,

string p2Names[],int SIZE7,string names[],int SIZE8, vector<string> &vNames){

do {

cout << endl << p1Name << " your game has successfully loaded. \n"

<< "Press 1: to see a summary of the files that were read.\n"

<< "Press 2: to confirm Guess[][] filled correctly\n"

<< "Press 3: to confirm my game board[][] was read the data in correctly\n"

<< " and to that the guess[][] were randomly filled.\n"

<< "Press 4: to start your game.\n"

<< "Press 5: to run Top Player's Board.\n"

<< "Press 6: to exit.\n";

cin >> ch;

switch(ch){

case '1': fileSum(count1,count2,numShp1,numShp2); break;

void cGssArr(const char [][9], const char [][9],const int,const int,int, int); // confirms guess[][] filled correctly

case '2': cGssArr(guessP1, guessP2,ROWS,COLS,p1GShps,p2GShps);break;

case '3': print2DArr(guessP1, guessP2,board1,board2,ROWS,COLS,maxGms,winner); break;

case '5': topPlyrs(p2Names,SIZE7,names,SIZE8,p1Name,vNames);

case '6': exit(EXIT\_SUCCESS); break;

default: cout << setw(9)<< " " << "Loading.......\n";

}

if(ch=='1' || ch=='2' || ch=='3'|| ch=='5'){

cout << "Run Menu again?\n";

cin >> ans;

} else ans='n';

// continue doing all the statements above until ans does not equal y or Y

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

}

// validates user's input

bool isReady(char ans){

// conditional validates user's input

if(ans=='y'){

return true;

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

return true;

} else return false;

}

// display instructions to player

void instruc(string player, const int MAX, int min){

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

<< setw(6) << " " << "your opponent\'s ship." << endl

<< endl << setw(12)<< " " << player << endl

<< setw(2) << " " << "Guess a letter between A-H\n"

<< setw(2) << " " << "and a number between " << min << " and " << MAX/8 << endl;

}

// display hit message when player guesses correctly

void hitMiss(bool isHit, int guess, int shipLoc){

cout << setw(13) << static\_cast<char>(guess) << " == " << static\_cast<char>(shipLoc) << endl;

if(isHit) cout << setw(20) << "It\'s a HIT!\n" << endl;

else cout << setw(22) << "It\'s a MISS!\n";

}

// display try again message when both player's guessed wrong

void hitMiss(){

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

}

// get player 1's name

void getName(string &name1){

cout << "\nPlayer 1: Enter your name ";

cin >> name1;

cout << endl;

upper(name1); // call function to convert user input into capital letters

}

// randomly fill guess[][] until at least one of the player's has 3 ships in their array

void fllGArr(char guessP1[][9],char guessP2[][9],char choices[],int ROWS,int COLS,int &p1GShps, int &p2GShps){

int size=17;

p1GShps=p2GShps=0; // initialize both players number of ships to zero

bool minMet; // minimum number of ships==3

do{

minMet=false; // set flag

for(int gRow=1; gRow<ROWS; gRow++){

for(int gCol=1; gCol<COLS; gCol++){

// automatically set player 1's guess[][] randomly from choices[]

guessP1[gRow][gCol]=choices[rand()%size]; // saves either a 'S' or 'B'

// track how many ships player 1's array has

if(guessP1[gRow][gCol]=='S'){

p1GShps++;

if(p1GShps==3) minMet=true; // reassign value to flag

}

// automatically set player 2's guess[][] randomly from choices[]

guessP2[gRow][gCol]=choices[rand()%size];

// track how many ships player 1's array has

if(guessP2[gRow][gCol]=='S'){

p2GShps++;

if(p2GShps==3) minMet=true; // reassign value to flag

}

}

}

} while(!minMet);

}

// confirm data that was read in is even and contains at least 3 ships

void fileSum(int count1, int count2, int numShp1, int numShp2){

cout << endl << setw(3)<<" " << "Read in " << setw(6)<< " " << "P1 Board" << setw(5)<< " "<< "P2 Board" << endl;

cout << "Total # chars " << setw(8) << right << count1 << "\t \t" << count2 << "\n";

cout << "Total # ships " << setw(8) << right << numShp1 << "\t \t" << numShp2 << "\n\n";

}

// save player 1 and all of player 2's names to a new array

void copyAdd(string p2Names[],const int SIZE7, string names[], const int SIZE8, string p1Name){

// copy all of player 2's name into a new array

for(int i=0;i<SIZE7;i++){

names[i]=p2Names[i];

}

// calculate the last index for names[]

unsigned int last = SIZE8-1;

// add player 1's name to the end of the names[]

names[last]=p1Name;

}

//

void chckArr(char &ans){

cout << "Would you like to confirm it was a hit? ";

cin >> ans;

}

// display data from both player's guess arrays.

void cGssArr(const char guessP1[][9], const char guessP2[][9],

const int ROWS,const int COLS, int p1GShps, int p2GShps){

cout << "\nConfirming Guess arrays are random and have at least 3 S in one array\n";

cout << setw(4) << " " << "P1 Guesses"<<endl;

prntArr(guessP1,ROWS,COLS);

cout <<"\nP1 # Ships : " << p1GShps <<endl<<endl;

cout << setw(4) << " " << "P2 Guesses\n";

prntArr(guessP2,ROWS,COLS);

cout << endl << "P2 # Ships : " << p2GShps << endl<<endl;

}

// binary search

int binarySrch(string names[], string &name, int SIZE8){

int indx=0,

first=0,

last=SIZE8-1,

middle,

position=-1;

bool found=false;

upper(name); // change string to all uppercase letters

while(!found && first<=last){ // search between indices [0,7]

middle = (first + last)/2; // middle index

if(names[middle]==name){ // check if middle indx equals name

found=true;

position=middle;

// if name ASCii value is smaller then its in lower half of array

} else if(names[middle]>name){

last= middle-1;

// if name ASCii value is smaller then its in upper half of array

} else first = middle+1;

}

return position;

}

// displays game's name and instructions in a banner

void banner(string str){

// reassign variables value

str="BATTLESHIP";

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

if(i==0 || i==2){

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

cout << "\*";

} cout << endl;

} else cout << setw(21) << str << endl;

}

}