BATTLESHIP!

Kristopher Schall

CIS 5

Spring 2017

44187

Table of Contents

Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 3

How the Game Works \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 3

My Approach to the Game \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 4

References\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 4

Proof of Working Product\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 5

Cross Reference\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 6

Program \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pg. 7

INTRODUCTION

Battleship has been around longer than most modern day board games. The first version of the game was played in the 1930s, using pads of paper with a pencil. The Board Game company Milton Bradley ultimately made a board version of the game in 1967. Throughout the years there has been different versions of the game including digital versions. Battleship was also created into a Film in 2012, nearly a century after the original game was created. Battleship is a must have for anyone’s game collection.1

How the Game Works

Battleship is most commonly played between two individuals, although some other versions require a different number of participants. The version that I have created is the standard two-player game. The setup of the game consists of the two players placing their ships on a 10 by 10 grid. Each player has a total of five ships to place. They are: Destroyer (2 spaces) , Cruiser (3 spaces), Submarine (3 Spaces), Carrier (5 Spaces), and the games Namesake The BATTLESHIP! (4 spaces) After the players have placed their ships on their aquatic battlefield The Battle Begins! Each player takes turns to launch missiles in hopes of sinking their opponents ships. Each player must state if it is a miss or a hit. After each ship is destroyed it must be stated that it was sunk. After the last ship is sunk it is tradition to state “You Sunk my Battleship!” The player with ships still afloat wins the game.

My Approach towards the Game

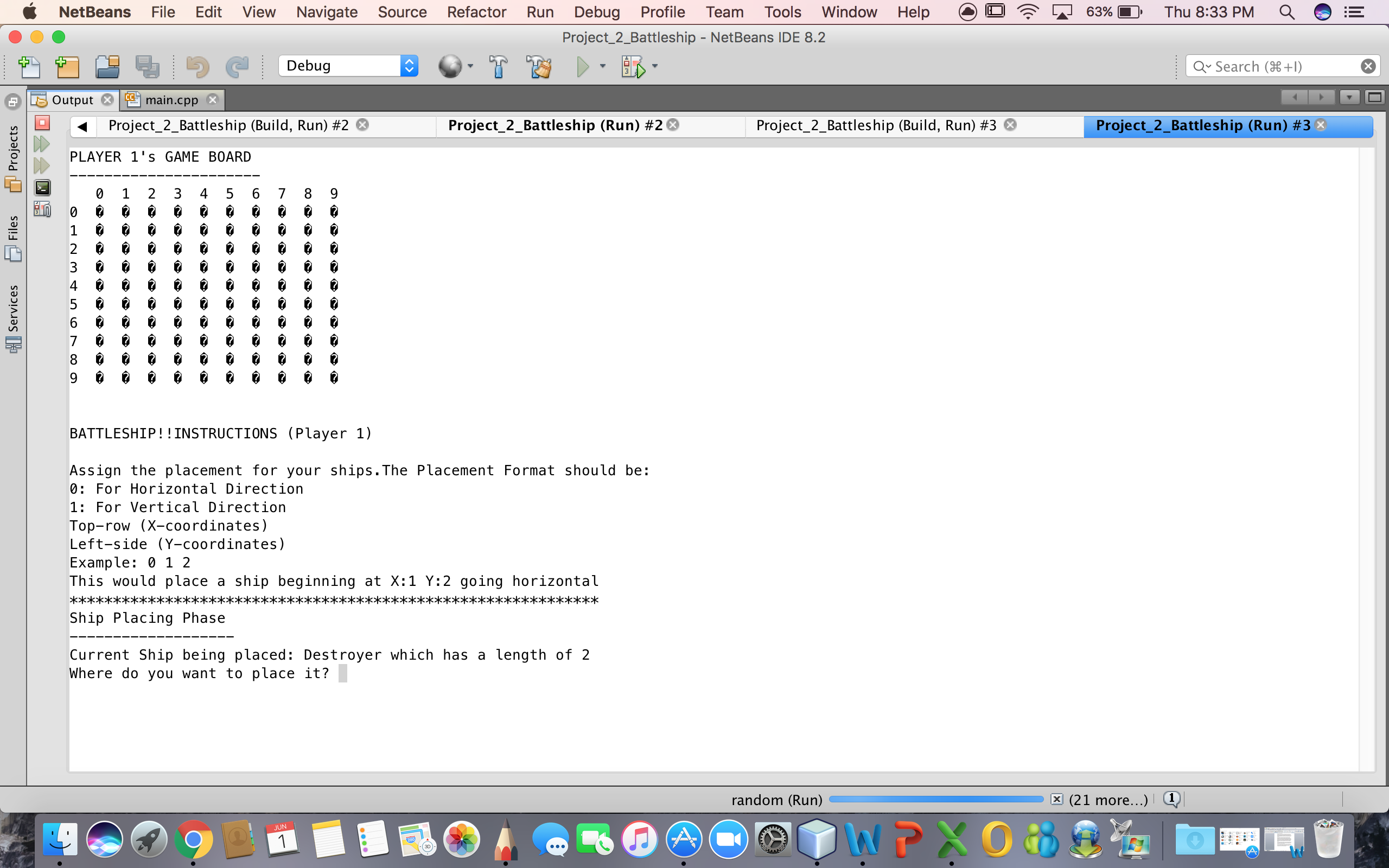
My initial approach towards the game was to keep it as close to the original as possible. Being a fan of the game my whole life, the original is always the best. Overall the game is simple and does not take too much effort for two players to have a quick game. It may not be as streamlined as I would like, but I feel the game I have created pays homage to the game of Battleship.

References

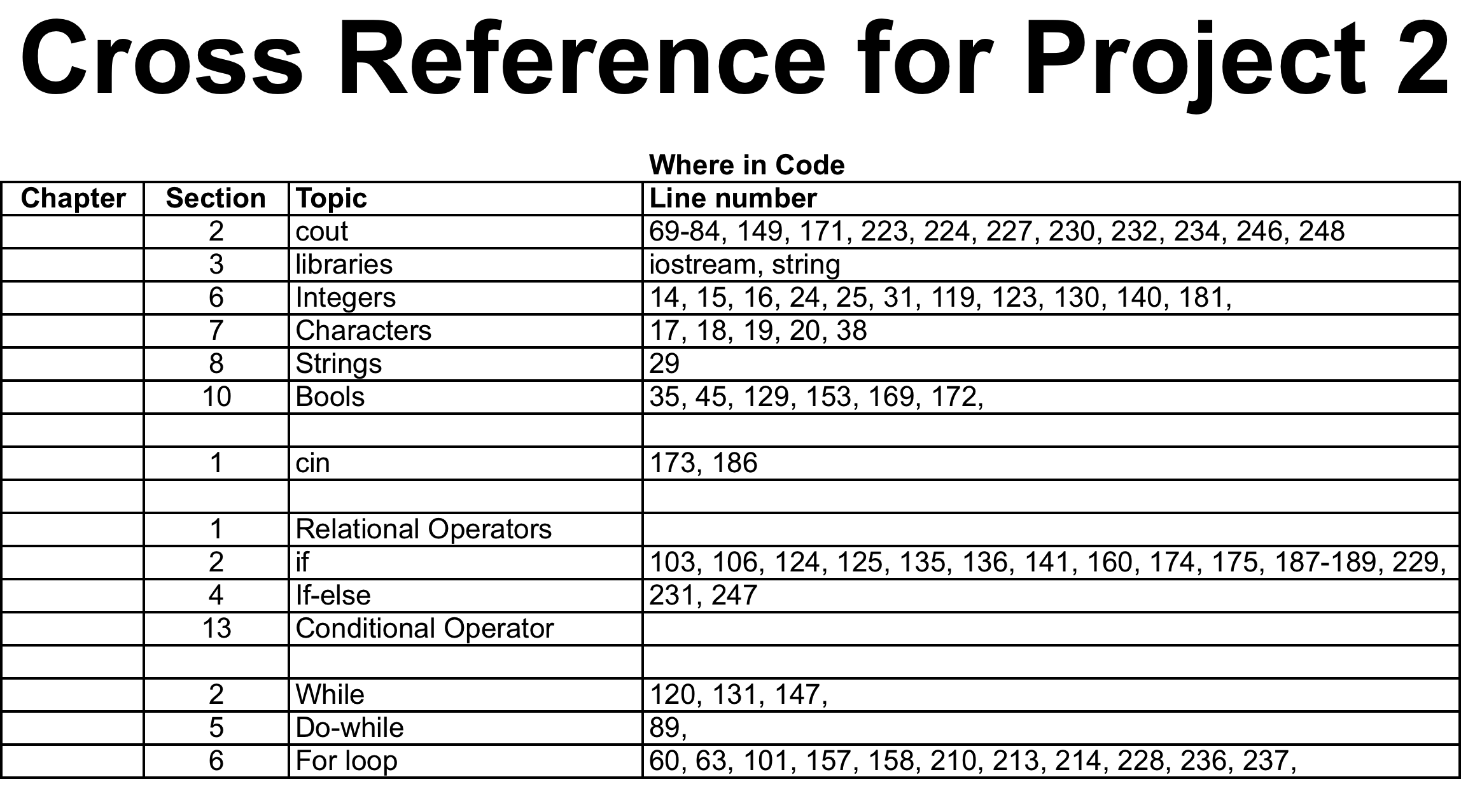
Dr. Lehr’s Lectures & Lab

“Starting Out with C++: From Control Structures through Objects” Gaddis, Tony. 8th Edition. (Textbook)

Working Product

****

Cross Reference



Program Code

#include <iostream>

#include <string>

using namespace std;

const int BOARD\_WIDTH = 10;

const int BOARD\_HEIGHT = 10;

const int SHIP\_TYPES = 5;

const char H20 = 247; //ASCII Character Code

const char HitShip = 'X';

const char ShipPlace = 'S';

const char MissShip = '0';

struct POINT {

//A location on the grid defined

//by X(horizontal) Y(vertical) coordinates

int X;

int Y;

};

struct SHIP {

//Ship name

string name;

//Total points on the grid

int length;

//Coordinates of those points

POINT onGrid[5]; //0-4 max length of biggest ship

//Whether or not those points are a "hit"

bool hitFlag[5];

}ship[SHIP\_TYPES];

struct PLAYER {

char grid[BOARD\_WIDTH][BOARD\_HEIGHT];

}player[3]; //Ignore player 0, just using player's 1 & 2

enum DIRECTION {HORIZONTAL,VERTICAL};

struct PLACESHIPS {

DIRECTION direction;

SHIP shipType;

};

bool gameRunning = false;

//Functions

void LoadShips();

void ResetBoard();

void DrawBoard(int);

PLACESHIPS UserInputShipPlacement();

bool UserInputAttack(int&,int&,int);

bool GameOverCheck(int);

int main()

{

LoadShips();

ResetBoard();

//"Placing Ships" phase of game

for (int aplyr=1; aplyr<3; ++aplyr)

{

//Loop through each ship type to place

for (int thisShip=0; thisShip<SHIP\_TYPES; ++thisShip)

{

//Display gameboard for player

DrawBoard(aplyr);

//Gameplay instructions

cout<<"\n\nBATTLESHIP!!";

cout<<"INSTRUCTIONS (Player " << aplyr << ")\n\n";

cout<<"Assign the placement for your ships.";

cout<<"The Placement Format should be:\n";

cout<<"0: For Horizontal Direction\n";

cout<<"1: For Vertical Direction\n";

cout<<"Top-row (X-coordinates)\n";

cout<<"Left-side (Y-coordinates)\n";

cout<<"Example: 0 1 2\n";

cout<<"This would place a ship beginning at X:1 Y:2 going horizontal\n";

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"Ship Placing Phase\n";

cout<<"-------------------\n";

cout<<"Current Ship being placed: " << ship[thisShip].name

<< " which has a length of " << ship[thisShip].length << "\n";

cout<<"Where do you want to place it? ";

//Get input from user and loop until good data is returned

PLACESHIPS aShip;

aShip.shipType.onGrid[0].X = -1;

while (aShip.shipType.onGrid[0].X == -1)

{

aShip = UserInputShipPlacement();

}

//Combine user data with "this ship" data

aShip.shipType.length = ship[thisShip].length;

aShip.shipType.name = ship[thisShip].name;

//Add the FIRST grid point to the current player's game board

player[aplyr].grid[aShip.shipType.onGrid[0].X][aShip.shipType.onGrid[0].Y] = ShipPlace;

//Determine ALL grid points based on length and direction

for (int i=1; i<aShip.shipType.length; ++i)

{

if (aShip.direction == HORIZONTAL){

aShip.shipType.onGrid[i].X = aShip.shipType.onGrid[i-1].X+1;

aShip.shipType.onGrid[i].Y = aShip.shipType.onGrid[i-1].Y; }

if (aShip.direction == VERTICAL){

aShip.shipType.onGrid[i].Y = aShip.shipType.onGrid[i-1].Y+1;

aShip.shipType.onGrid[i].X = aShip.shipType.onGrid[i-1].X; }

//Add the REMAINING grid points to our current players game board

player[aplyr].grid[aShip.shipType.onGrid[i].X][aShip.shipType.onGrid[i].Y] = ShipPlace;

}

//Loop back through each ship type

}

//Loop back through each player

}

//Game Playing Phase

gameRunning = true;

int thisPlayer = 1;

do {

//Because we are ATTACKING now, the

//opposite players board is the display board

int enemyPlayer;

if (thisPlayer == 1) enemyPlayer = 2;

if (thisPlayer == 2) enemyPlayer = 1;

system("cls");

DrawBoard(enemyPlayer);

//Get attack coords from this player

bool goodInput = false;

int x,y;

while (goodInput == false) {

goodInput = UserInputAttack(x,y,thisPlayer);

}

//Check board; if a ship is there, set as HIT.. otherwise MISS

if (player[enemyPlayer].grid[x][y] == ShipPlace) player[enemyPlayer].grid[x][y] = HitShip;

if (player[enemyPlayer].grid[x][y] == H20) player[enemyPlayer].grid[x][y] = MissShip;

//Check to see if the game is over

//If 0 is returned, nobody has won yet

int aWin = GameOverCheck(enemyPlayer);

if (aWin != 0) {

gameRunning = false;

break;

}

//Alternate between each player as we loop back around

thisPlayer = (thisPlayer == 1) ? 2 : 1;

} while (gameRunning);

system("cls");

cout << "\n\nYOU SUNK MY BATTLESHIP!!! PLAYER " << thisPlayer << " HAS WON THE GAME!\n\n\n\n";

system("pause");

return 0;

}

bool GameOverCheck(int enemyPLAYER)

{

bool winner = true;

//Loop through enemy board

for (int w=0; w<BOARD\_WIDTH; ++w){

for (int h=0; h<BOARD\_HEIGHT; ++h){

//If any ships remain, game is NOT over

if (player[enemyPLAYER].grid[w][h] = ShipPlace)

{

winner = false;

return winner;

}

}}

//If we get here, somebody won, game over!

return winner;

}

bool UserInputAttack(int& x, int& y, int theplayer)

{

cout << "\nPLAYER " << theplayer << ", ENTER COORDINATES TO ATTACK: ";

bool goodInput = false;

cin >> x >> y;

if (x<0 || x>=BOARD\_WIDTH) return goodInput;

if (y<0 || y>=BOARD\_HEIGHT) return goodInput;

goodInput = true;

return goodInput;

}

PLACESHIPS UserInputShipPlacement()

{

int d, x, y;

PLACESHIPS tmp;

//Using this as a bad return

tmp.shipType.onGrid[0].X = -1;

//Get 3 integers from user

cin >> d >> x >> y;

if (d!=0 && d!=1) return tmp;

if (x<0 || x>=BOARD\_WIDTH) return tmp;

if (y<0 || y>=BOARD\_HEIGHT) return tmp;

//Good data

tmp.direction = (DIRECTION)d;

tmp.shipType.onGrid[0].X = x;

tmp.shipType.onGrid[0].Y = y;

return tmp;

}

void LoadShips()

{

//Sets the default data for the ships

//we plan to include in the game

//IMPORTANT!! > MUST MATCH SHIP\_TYPES -Default=5 (0-4)

ship[0].name = "Destroyer"; ship[0].length = 2;

ship[1].name = "Cruiser"; ship[1].length = 3;

ship[2].name = "Submarine"; ship[2].length = 3;

ship[3].name = "Battleship"; ship[3].length = 4;

ship[4].name = "Carrier"; ship[4].length = 5;

}

void ResetBoard()

{

//Loop through each player

for (int plyr=1; plyr<3; ++plyr)

{

//For each grid point, set contents to 'water'

for (int w=0; w<BOARD\_WIDTH; ++w){

for (int h=0; h<BOARD\_HEIGHT; ++h){

player[plyr].grid[w][h] = H20;

}}

//Loop back to next player

}

}

void DrawBoard(int thisPlayer)

{

//Draws the board for a player (thisPlayer)

cout << "PLAYER " << thisPlayer << "'s GAME BOARD\n";

cout << "----------------------\n";

//Loop through top row (board\_width) and number columns

cout << " ";

for (int w=0; w<BOARD\_WIDTH; ++w) {

if (w < 10)

cout << w << " ";

else if (w >= 10)

cout << w << " ";

}

cout << "\n";

//Loop through each grid point and display to console

for (int h=0; h<BOARD\_HEIGHT; ++h){

for (int w=0; w<BOARD\_WIDTH; ++w){

//If this is the FIRST (left) grid point, number the grid first

if (w==0) cout << h << " ";

//If h was 1 character long, add an extra space to keep numbers lined up

if (w<10 && w==0) cout << " ";

//Display contents of this grid (if game isn't running yet, we are placing ships

if (gameRunning == false) cout << player[thisPlayer].grid[w][h] << " ";

//Do not show ships, show damage if it's hit instead

if (gameRunning == true && player[thisPlayer].grid[w][h] != ShipPlace)

{cout << player[thisPlayer].grid[w][h] << " ";}

else if (gameRunning == true && player[thisPlayer].grid[w][h] == ShipPlace)

{cout << H20 << " ";}

if (w == BOARD\_WIDTH-1) cout << "\n";

}

}

}