Project 2

Title

ASCII BattleShip!
An ASCII Battleship Simulation

Course

CSC-17A

Section

42636

Due Date

June 1, 2017

Author

Scott Parker

1 Introduction

BattleShip is a classic board game that has multiple layers of complexity. This simulation features 5 ships of varying sizes: BattleShip which takes 5 hits to sink, Destroyer which takes 4 hits to sink, Submarine which takes 3 hits to sink and a PT Boat which takes 2 hits to sink for a total of 14 hits to sink the entire fleet.

2 Game Play and Rules

The game is played by first placing 5 ships into the ocean. The ocean is made up of a 2D array of 10 x 10 lines with each row and column representing an x,y coordinate pair.

After the first player places their ships then the second player (computer player) will place all available ships. Players then alternate guesses in an attempt to sink the other players fleet. The victory conditions occur when one player first scores 14 hits (sinks the entire enemy fleet).

The player is given the opportunity to save or exit the game at the end of each round, which if chosen as an option will send the game back to the initial loading screen.

3 Development Summary

Lines of Code	**~1026**
Dedicated Comment Lines	**~150**
Blank Lines (White space)	**~50**
Total Lines of Source Files	**800+**

The project was completely coded from scratch. All of the code, headers files, functions, and features were 100% developed in-house without copying any external source code. The project took approximately 45 hours to develop over a 9 day period (not including documentation) not inclusive of the original BattleShip Project 1 upon which this project is based. Very limited characteristics were brought over from the original project.

3.1 Comments on Development

Working with 2D arrays in a structure was a challenging objective and eventually I found no other way than to use literals as the size definitions for the arrays. It would have been easier to use dynamic arrays and the use of literals could have been avoided, however due to time constraints rewriting the entire project to accommodate this was not feasible while still completing the project on time and still implementing the new programming features required.

Saving and restoring games is a feature which was interesting to implement. The first aspect is the bones file which holds a list of valid save-game files. The second feature is the binary save file which uses the name of the save file. Duplicate file names overwrite existing files and the size of the bones file is dynamic string array managed and checked to avoid file-creep (excessive blank lines or non-existent files names added at the end or beginning of the file).

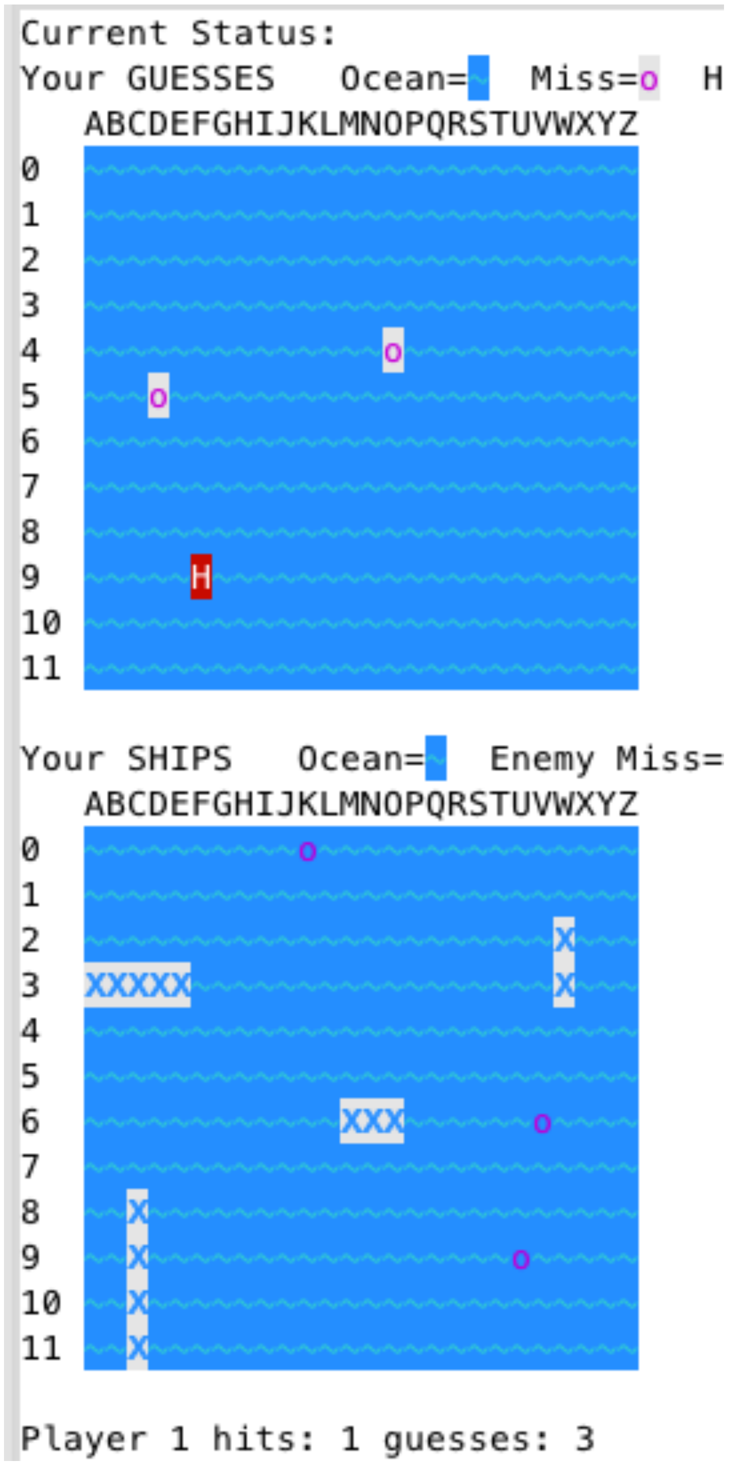
3.1.1 Display Elements

Even though this is a text-based ASCII program the games does feature an extensive color element. It uses ANSI color codes and is known to work correctly on Windows 10 and Macintosh systems. The ANSI formatting is designed to mimic a dedicated video game window rather than being a simply being a text element in the console.

3.1.2 New Features from added since Project 1

The main upgrade from Project 1 is the change of the map size to a 10 x 10 grid. This significantly reduces the number of turns needed to complete the game. As a result of this the cheat mode was removed and the computer no longer receives bonus free hits. The graphic and layout has also been significantly modified. The previous version featured two top-over-bottom maps which were quite large and unwieldy. The final version has two side-by-side maps which are much easier and intuitive to navigate as well as having a significantly more striking visual impact as evidenced by the screenshots below:

Project 1 Map Image

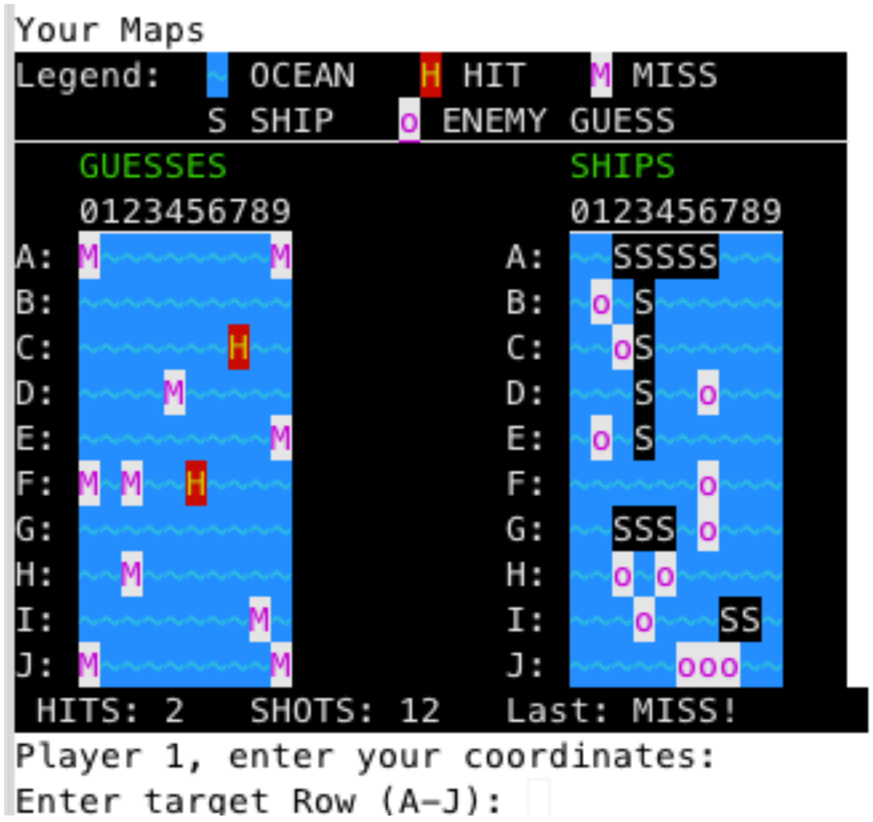


```
Current Status:
Your GUESSES   Ocean=~  Miss=o  H
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
0  ~~~~~~~~~~~~~~~~~~~~~~~~~~
1  ~~~~~~~~~~~~~~~~~~~~~~~~~~
2  ~~~~~~~~~~~~~~~~~~~~~~~~~~
3  ~~~~~~~~~~~~~~~~~~~~~~~~~~
4  ~~~~~~~~~~~~~~o~~~~~~~~~~~
5  ~~~o~~~~~~~~~~~~~~~~~~~~~~
6  ~~~~~~~~~~~~~~~~~~~~~~~~~~
7  ~~~~~~~~~~~~~~~~~~~~~~~~~~
8  ~~~~~~~~~~~~~~~~~~~~~~~~~~
9  ~~~~~H~~~~~~~~~~~~~~~~~~~~
10 ~~~~~~~~~~~~~~~~~~~~~~~~~~
11 ~~~~~~~~~~~~~~~~~~~~~~~~~~

Your SHIPS   Ocean=~  Enemy Miss=
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
0  ~~~~~~~~~~o~~~~~~~~~~~~~~~
1  ~~~~~~~~~~~~~~~~~~~~~~~~~~
2  ~~~~~~~~~~~~~~~~~~~~~~X~~~
3  XXXXX~~~~~~~~~~~~~~~~~X~~~
4  ~~~~~~~~~~~~~~~~~~~~~~~~~~
5  ~~~~~~~~~~~~~~~~~~~~~~~~~~
6  ~~~~~~~~~~~~XXX~~~~~~o~~~~
7  ~~~~~~~~~~~~~~~~~~~~~~~~~~
8  ~~X~~~~~~~~~~~~~~~~~~~~~~~
9  ~~X~~~~~~~~~~~~~~~~~o~~~~~
10 ~~X~~~~~~~~~~~~~~~~~~~~~~~
11 ~~X~~~~~~~~~~~~~~~~~~~~~~~

Player 1 hits: 1 guesses: 3
```

Project 2 Map Image



```
Your Maps
Legend:  ~ OCEAN   H HIT   M MISS
         S SHIP   o ENEMY GUESS
   GUESSES                SHIPS
   0123456789             0123456789
A: M~~~~~~~~M          A: ~~SSSSS~~~
B: ~~~~~~~~~~          B: ~o~S~~~~~~
C: ~~~~~~~H~~          C: ~~oS~~~~~~
D: ~~~~M~~~~~          D: ~~~S~~o~~~
E: ~~~~~~~~~M          E: ~o~S~~~~~~
F: M~M~~H~~~~          F: ~~~~~~o~~~
G: ~~~~~~~~~~          G: ~~SSS~o~~~
H: ~~M~~~~~~~          H: ~~o~o~~~~~
I: ~~~~~~~~M~          I: ~~~o~~~SS~
J: M~~~~~~~~M          J: ~~~~~ooo~~
 HITS: 2   SHOTS: 12   Last: MISS!
Player 1, enter your coordinates:
Enter target Row (A-J):
```

4 Features

4.1 Advanced Concepts Used

- Dynamic Arrays   used in main.cpp Line 343
- 2-dimensional Arrays used in Map.h Line 20, 21
- Reading Binary Files used in main.cpp Line 436, 437
- Writing Binary Files used in main.cpp Line 396, 397
- Reading Text Files used in main.cpp Line 349, 420
- Custom Structures used in Map.h file
- Structures containing arrays used in Map.h file
- Passing structure into function used in main.cpp Line 61, 62
- Boolean functions used in main.cpp Line 34
- Classes used in Game.h, Guess.h
- Inheritance used in Game.h
- Operator Overloading operators overloaded in Game.cpp line 130, Guess.cpp line 39, 50
- Polymorphism with derived class passed by reference to function line 72, 73
- Templates used in val.h line 110, 123

4.2 *Current Issues*

- To date there are still a few known unresolved issues with the project. Primarily these issues are:
    - Sometimes the ANSI formatting will change for unknown reasons.
    - Occasionally map data becomes corrupted (most frequently after changing OS systems and loading saved game files from the previous OS) and certain coordinate pairs return erroneous data (invalid choice for an ocean space)
    - When the “bizarre” game is saved it will crash when loaded due to writing derived data type to object that is expecting parent data type (writing ‘Weirdo’ as a ‘Game’ object)

5 References

Extensive information was researched on Google.
The most valuable repositories of information was from:

www.stackoverflow.com
www.cplusplus.com

6 Code and Pseudocode

6.1 *Pseudocode*

```
/*
 * File:   main.cpp
 * Author: Scott Parker
 * Created on May 20, 2017, 11:30 AM
 * Purpose: Battleship game Project 2
 * Notes: Revised version for Project 2 to include features from later
 * chapters
 */

//System Libraries
#include <iostream>
#include <cctype>
#include <ctime>
#include <cstdlib>
#include <fstream>
using namespace std;

//User Libraries
#include "colors.h"
#include "val.h"
#include "Game.h"
#include "Map.h"
#include "Guess.h"


//Global Constants
//Such as PI, Vc, -> Math/Science values
//as well as conversions from one system of measurements to another
```

```
//Function Prototypes
//Start a new game
//Places the ships on the game board
//Check to see if ship can be put in location
//Place the computer ships
//Function to save the game
//Function to load a saved game
//Function to resume playing a saved game

//Executable code begins here! Always begins in Main
   //Set random seed
//Declare Variables
//Game menu
      //Output switch menu screen
      //enter 1 to resume saved game
      //enter 2 for a new game
      //0 or unlisted number to exit
      //Using template for data input and validation

      //Switch to determine the Problem
         //start a new game
         //Function to load a saved game
         //default option - exit menu
   //show menu while choices all active
   //Exit stage right! - This is the 'return 0' call

//************************   locTest   ****************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567
89
//0000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //Returns true if location is bad
//** Inputs: column, row, size of ship, verticle or horiz, player object
//** Outputs: return true if a bad location is encountered
//*****************************************************************************
bool locTest(int putRow, int putCol, int size, char verHor, Game &p) {
   //check orientation
      //Check to see if off map for vertical placement
         //Loop through size to check placement for overlap
      //Check to see if off map for horizontal placement
         //Loop through size to check placement for overlap
   //return OK or not
//************************   putShip   ****************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567
89
//0000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //function to place human player's ships
//** Inputs: Game p1 (player 1)
//** Outputs: none - will only manipulate the classes
```

```
//****************************************************************************
    //output the map to see current ship placement
    //loop through the ships
        //repeat until ship placement is valid
            //Check to see if ship can go here
        //place the ship
        //show ship placement
//*************************    putComp    ****************************************
//234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //function to place computer's ships
//** Inputs: Game p2 (player 2)
//** Outputs: none - will only manipulate the classes
//****************************************************************************
    //place computer ships on board
        //repeat until ship successful placed
            //test to see if ship successfully placed
//*************************    newBord    ****************************************
//234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //function to start a new game (new board)
//** Inputs: Game p1 (player 1), Game p2 (player 2)
//** Outputs: none - will only manipulate the classes
//****************************************************************************
    //reset player 1 to defaults
    //reset player 2 to defaults
    //place human player ships
    //Keep going to play game until end, exit or save
        //Enter guess
        //repeat until acceptable guess
        //Computer gets final shot even if all ships sank
        //Continue, save, or exit
            //function to save the game
            //exit without saving
            //Game lost or won due to fleet sank
//*************************    conBord    ****************************************
//234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //function to resume a saved game (continue board)
//** Inputs: Game p1 (player 1), Game p2 (player 2)
//** Outputs: none - will only manipulate the classes
//****************************************************************************
    //load a saved game
    //Keep going to play game until end, exit or save
        //Repeat until valid guess entered
        //update maps
        //Computer gets final shot even if all ships sank
        //Continue, save, or exit
           //function to save the game
           //exit without saving
           //Game lost or won due to fleet sank

//**********************   savGame   ******************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789
//0000000011111111122222222223333333333444444444455555555556666666666777777
777
//** Purpose:  //Function to save the game
//** Inputs: Game p1, Game p2;
//** Outputs: none
//******************************************************************************
void savGame(Game p1, Game p2) {
    //hold file names from bones file. also used as temp to hold names while counting lines
    //array of strings to hold file names
    //count variable and temp variable
    //create filestream object
    //open bones file in input mode
    //output error if file unable to open
    //otherwise
        //find the number of lines in the file
        //get file line, increment to next line
        //increment counter of the number of lines
    //close file
    //create dynamic array to hold bones file data
    //open bones file in input mode
    //output error if file fails to open
    //otherwise
        //loop through file line by line
            //get file line, increment to next array element
    //close file
    //loop through array line by line
        //display array element if not empty line
    //enter the name of new savefile
    //open bones file in output mode
    //output error message if file fails to open
    //otherwise
      //loop through array line by line
            //if line is not blank
                //output array element as line in file
    //close file
    //clean up memory and delete dynamic array
    //open binary file in output mode to save game data
    //write player 1 ship data to file
    //write player 1 guess data to file
    //close file
//************************   lodGame   ****************************************
```

```
//23456789012345678901234567890123456789012345678901234567890123456
789
//00000000111111111122222222223333333333444444444455555555556666666666777777
777
//** Purpose:  //Function to load a saved game
//** Inputs: Game &p1, Game &p2
//** Outputs: none
//****************************************************************************
    //hold file names from bones file. also used as temp to hold names while counting lines
    //create filestream object
    //reset player 1 to defaults
    //reset player 2 to defaults
    //open bones file in input mode
    //output error if file did not open correctly
        //loop through file line by line
            //get file line, increment to next line
            //output text from that line in file
    //close file
    //enter name of savefile to open
    //open binary file name provided
        //randomly place player 1 ships
        //randomly place player 2 ships
    //read in binary file data to populate game array data
        //read file to fill player 1 ship data
        //read file to fill player 1 guess data

/*
 * File:   Guess.h
 * Author: scott_r_parker
 *
 * Created on May 24, 2017, 4:54 PM
 */
#ifndef GUESS_H
#define GUESS_H
#include <iostream>
using namespace std;
// Forward Declaration
// Function Prototypes for Overloaded Stream Operators
class Guess{
//private data members
//public methods
    //set the row value manually
    //set the column value manually
    //return the row guess
    //Return the column guess
   // Friends
    //ostream overload
    //istream overload
#endif /* GUESS_H */

/*
```

```
 * File:   Player.h
 * Author: scott_r_parker
 *
 * Created on April 13, 2017, 2:36 PM
 */
#ifndef MAP_H
#define MAP_H
      //Map to keep track of guesses (hits and misses)
      //Map to keep track of player's ships (undamaged, placements, and hit)
#endif /* MAP_H */

/*
 * File:   val.h
 * Author: scott_r_parker
 *
 * Created on May 22, 2017, 6:57 PM
 */

 /* //input and validation functions.*/


#ifndef VAL_H
#define VAL_H

#include <iostream>
#include <cctype>
using namespace std;

//create namespace

//input and validate character (char) data.
    //Up to 4 inputs, each input is an acceptable value for the return value
    //of the function...
    //choose letter from LOW to HIGH
    //enter a char (up to 5 valid choices)
    //input and validate a signed int
    //first input is minimum value, 2nd input is max value
    //Overloaded (no arguments) only validates input TYPE
    //long long int for low/high range
#endif /* VAL_H */
```

6.2 Source Code

```
/*
 * File:   main.cpp
 * Author: Scott Parker
 * Created on May 20, 2017, 11:30 AM
 * Purpose: Battleship game Project 2
 * Notes: Revised version for Project 2 to include features from later
 * chapters
 */

//System Libraries
#include <iostream>
#include <cctype>
#include <ctime>
#include <cstdlib>
#include <fstream>
using namespace std;

//User Libraries
#include "colors.h"
#include "val.h"
#include "Game.h"
#include "Map.h"
#include "Guess.h"


//Global Constants
//Such as PI, Vc, -> Math/Science values
//as well as conversions from one system of measurements to another

//Function Prototypes
void newBord(Game &, Game &); //Start a new game
void putShip(Game &); //Places the ships on the game board
bool locTest(int, int, int, char, Game &); //Check to see if ship can be put in location
void putComp(Game &); //Place the computer ships
void savGame(Game, Game); //Function to save the game
void lodGame(Game &, Game &); //Function to load a saved game
void conBord(Game &, Game &); //Function to resume playing a saved game

//Executable code begins here! Always begins in Main
int main(int argc, char** argv) {
    //Set random seed
    srand(static_cast<unsigned int>(time(0)));

    //Declare Variables
    int choice=0;
    Game p1; //Player 1
    Game p2; //Player 1

    //Game menu
    do{
        //Output switch menu screen
        cout<<"Choose from the list <non-numeric data will be ignored>"<<endl;
        cout<<"Enter 1 to resume a saved game"<<endl; //enter 1 to resume saved game
        cout<<"Enter 2 to start a new game"<<endl; //enter 2 for a new game
        cout<<"Enter 0 (zero) or a number not listed to exit."<<endl; //0 or unlisted number to exit
        choice=val::inNum(choice); //Using template for data input and validation

        //Switch to determine the Problem
        switch(choice){
            case 2:{newBord(p1, p2);break;} //start a new game
            case 1:{conBord(p1, p2);break;} //Function to load a saved game
            default:
                cout<<"You are exiting the game"<<endl; //default option - exit menu
        }
    }while(choice>=1&&choice<=2); //show menu while choices all active

    //Exit stage right! - This is the 'return 0' call
    return 0;
}

//************************   locTest   ****************************************
//23456789012345678901234567890123456789012345678901234567890123456
789
//00000000111111111122222222223333333333444444444455555555556666666666777777
777
//** Purpose:  //Returns true if location is bad
//** Inputs: column, row, size of ship, verticle or horiz, player object
//** Outputs: return true if a bad location is encountered
//******************************************************************************
bool locTest(int putRow, int putCol, int size, char verHor, Game &p) {
    bool retVal=true;
    if (verHor=='v' || verHor=='V') {
        if (putRow+size>10) {
            cout<<"Ship is off the world!"<<endl;
            retVal=true;
        } else {
            cout<<"Rows within range for vertical placement."<<endl;
            cout<<"Checking overlap!"<<endl;
            for (int i=0;i<size;i++) {
                if (p.getShip(putRow+i, putCol)=='~') {
                    retVal=false;
                } else {
                    retVal=true;
                    cout<<"Ship collision!"<<endl;
                    break;
                }
            }
        }
    } else {
        if (putCol+size>10) {
            cout<<"Ship is off the world!"<<endl;
            retVal=true;
        } else {
            cout<<"Rows within range for horizontal placement."<<endl;
            cout<<"Checking overlap!"<<endl;
            for (int i=0;i<size;i++) {
                if (p.getShip(putRow, putCol+i)=='~') {
                    retVal=false;
                } else {
                    retVal=true;
                    cout<<"Ship collision!"<<endl;
                    break;
                }
            }
        }
    }
    return retVal;
}

//************************   putShip   ***************************************
//234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000001111111111222222222233333333334444444444555555555566666666667777777
777
//** Purpose:  //function to place human player's ships
//** Inputs: Game p1 (player 1)
//** Outputs: none - will only manipulate the classes
//****************************************************************************
void putShip(Game &p) {
    int putRow=0, putCol=0;
    char verHor='\0';
    cout<<p<<endl;
    cout<<"Player 1, place your ships!"<<endl;
    for (int i=5;i>=2;i--) {
        cout<<"Place your ";
        if (i==5) cout<<"Battleship!"<<endl;
        else if (i==4) cout<<"Destroyer!"<<endl;
        else if (i==3) cout<<"Submarine!"<<endl;
        else cout<<"PT Boat!"<<endl;
        bool badLoc=true;
        while (badLoc) {
            cout<<"Choose the row to place the bow of the ship: "<<endl;
            char tmprow=val::inAlpha('A', 'J');
            tmprow=toupper(tmprow);
            putRow=tmprow-65;
            cout<<"choose the column to place the bow of the ship: "<<endl;
            putCol=val::inNum(putRow, 0, 9);
            cout<<"Enter V to place Vertially or H to place Horizontally!"<<endl;
            verHor=val::inChar('v', 'V', 'h', 'H');
            verHor=toupper(verHor);
            badLoc=locTest(putRow, putCol, i, verHor, p);
        }
        p.setShip(putRow, putCol, i, verHor);
        cout<<p<<endl;
    }
}
```

```
//************************   putComp   ***************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000001111111111222222222233333333334444444444555555555566666666667777777
777
//** Purpose:  //function to place computer's ships
//** Inputs: Game p2 (player 2)
//** Outputs: none - will only manipulate the classes
//**************************************************************************
void putComp(Game &p) {
   int putRow=0, putCol=0;
   char verHor='\0';
   for (int i=5;i>=2;i--) {
      cout<<"Placing computer's ";
      if (i==5) cout<<"Battleship!"<<endl;
      else if (i==4) cout<<"Destroyer!"<<endl;
      else if (i==3) cout<<"Submarine!"<<endl;
      else cout<<"PT Boat!"<<endl;
      bool badLoc=true;
      while (badLoc) {
         putRow=rand()%10;
         putCol=rand()%10;
         if (rand()%2==0) {
            verHor='V';
         } else {
            verHor='H';
         }
         badLoc=locTest(putRow, putCol, i, verHor, p);
         badLoc?cout<<"Oops! Trying again!"<<endl:cout<<"Ship Placed!"<<endl;
      }
      p.setShip(putRow, putCol, i, verHor);
   }
}

//************************   newBord   ***************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000001111111111222222222233333333334444444444555555555566666666667777777
777
//** Purpose:  //function to start a new game (new board)
//** Inputs: Game p1 (player 1), Game p2 (player 2)
//** Outputs: none - will only manipulate the classes
//**************************************************************************
void newBord(Game &p1, Game &p2) {
   char chooser;
   Guess fire;
   p1.mapClr(); //reset player 1 to defaults
   p2.mapClr(); //reset player 2 to defaults
   putShip(p1); //place human player ships
   cout<<"Placing computer ships..."<<endl;
   putComp(p2);
```

```
    cout<<"Starting game..."<<endl;
    while (p1.gameOn() && p2.gameOn()) { //Keep going to play game until end, exit or save
        cout<<"Your Maps"<<endl;
        cout<<p1<<endl;
        do {
            cout<<"Player 1, enter your coordinates: "<<endl;
            cin>>fire;
            p2.okShot(fire);
            if (!p2.getShot()) {
                cout<<"Invalid! Try different coordinates!"<<endl;
            }
        } while (!p2.getShot());
        p1.setMap(fire, p2.getShip(fire.getRow(), fire.getCol()));
        cout<<"You "<<p1.getLast()<<endl;

        if (p1.getHits()==14) {
            cout<<"Computer taking final shot!"<<endl;
        } else {
            cout<<"Computer taking a shot!"<<endl;
        }
        do {
            int tmpRow=rand()%10;
            int tmpCol=rand()%10;
            fire.setRow(tmpRow);
            fire.setCol(tmpCol);
            p1.okShot(fire);
        } while (!p1.getShot());
        p2.setMap(fire, p1.getShip(fire.getRow(), fire.getCol()));
        cout<<"The computer's shot was a "<<p2.getLast()<<endl;
        cout<<"(C)ontinue, (S)ave, or e(X)it? "<<endl;
        chooser=val::inChar('c', 'C', 's', 'S', 'X');
        chooser=toupper(chooser);
        if (chooser=='S') {
            cout<<"Enter requested data to save the game"<<endl;
            savGame(p1, p2); //function to save the game
            p1.setPlay(chooser);
        } else if (chooser=='X') {
            cout<<"Exiting without saving!"<<endl;
            p1.setPlay(chooser);
        }
        if (p1.getHits()==14) {
            cout<<"You have sank all of the enemy ships!"<<endl;
            p1.setPlay('X');
        }
        else if (p2.getHits()==14) {
            cout<<"The enemy sank all of your ships!"<<endl;
            p2.setPlay('X');
        }
    }
}
```

```
//***********************    conBord    ***************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789
//0000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //function to resume a saved game (continue board)
//** Inputs: Game p1 (player 1), Game p2 (player 2)
//** Outputs: none - will only manipulate the classes
//**************************************************************************
void conBord(Game &p1, Game &p2) {
   char chooser;
   Guess fire;
   lodGame(p1, p2); //load a saved game
   while (p1.gameOn() && p2.gameOn()) { //Keep going to play game until end, exit or save
      cout<<"Your Maps"<<endl;
      cout<<p1<<endl;
      do {
         cout<<"Player 1, enter your coordinates: "<<endl;
         cin>>fire;
         p2.okShot(fire);
         if (!p2.getShot()) {
            cout<<"Invalid! Try different coordinates!"<<endl;
         }
      } while (!p2.getShot());
      p1.setMap(fire, p2.getShip(fire.getRow(), fire.getCol()));
      cout<<"You "<<p1.getLast()<<endl;

      if (p1.getHits()==14) {
         cout<<"Computer taking final shot!"<<endl;
      } else {
         cout<<"Computer taking a shot!"<<endl;
      }
      do {
         int tmpRow=rand()%10;
         int tmpCol=rand()%10;
         fire.setRow(tmpRow);
         fire.setCol(tmpCol);
         p1.okShot(fire);
      } while (!p1.getShot());
      p2.setMap(fire, p1.getShip(fire.getRow(), fire.getCol()));
      cout<<"The computer's shot was a "<<p2.getLast()<<endl;
      cout<<"(C)ontinue, (S)ave, or e(X)it? "<<endl;
      chooser=val::inChar('c', 'C', 's', 'S', 'X');
      chooser=toupper(chooser);
      if (chooser=='S') {
         cout<<"Enter requested data to save the game"<<endl;
         savGame(p1, p2); //function to save the game
         p1.setPlay(chooser);
      } else if (chooser=='X') {
         cout<<"Exiting without saving!"<<endl;
         p1.setPlay(chooser);
```

```
        }
        if (p1.getHits()==14) {
            cout<<"You have sank all of the enemy ships!"<<endl;
            p1.setPlay('X');
        }
        else if (p2.getHits()==14) {
            cout<<"The enemy sank all of your ships!"<<endl;
            p2.setPlay('X');
        }
    }
}

//**********************   savGame   ****************************************
//2345678901234567890123456789012345678901234567890123456789012345678901234567
89
//0000000011111111112222222222333333333344444444445555555555666666666677777777
777
//** Purpose:  //Function to save the game
//** Inputs: Game p1, Game p2;
//** Outputs: none
//****************************************************************************
void savGame(Game p1, Game p2) {
    string line=""; //hold file names from bones file. also used as temp to hold names while
counting lines
    string *gName=nullptr; //array of strings to hold file names
    int count=0, temp=0; //count variable and temp variable
    fstream fil; //create filestream object
    fil.open ("bones.txt", ios::in); //open bones file in input mode
    if (fil.fail()){ //output error if file unable to open
        cout<<"ERROR! Unable to open BONES file!"<<endl;
        cout<<BLUE<<BGGREEN<<"Creating BONES file!"<<RESET<<endl;
        fil.clear();
        fil.open ("bones.txt", ios::out);
        fil.close();
        fil.open("bones.txt", ios::in);
    } else { //otherwise
        while (fil){ //find the number of lines in the file
            getline(fil, line); //get file line, increment to next line
            count++; //increment counter of the number of lines
        }
    }
    fil.close(); //close file

    if (count<2) count=2;
    gName=new string [count]; //create dynamic array to hold bones file data
    fil.open ("bones.txt", ios::in); //open bones file in input mode
    if (fil.fail()){ //output error if file fails to open
        cout<<"ERROR! Unable to open bones file!"<<endl;
    } else { //otherwise
        for (int i=0;i<count;i++){ //loop through file line by line
            getline(fil, gName[i]); //get file line, increment to next array element
        }
    }
    fil.close(); //close file
    cout<<"Current Save Files"<<endl;
    for (int i=0;i<count;i++) { //loop through array line by line
        if (gName[i]!=""){ //display array element if not empty line
            cout<<"Save file name: "<<gName[i]<<endl;
        }
    }

    cout<<BOLDRED<<"WARNING! Duplicate names will overwrite!"<<RESET<<endl;
    cout<<BOLDRED<<"WARNING! Only enter alphanumeric characters!"<<RESET<<endl;
    cout<<"Enter the game name to save: "<<endl;
    cin.ignore(256, '\n');
    getline(cin, line); //enter the name of new savefile

    temp=0;
    while (gName[temp]!=line && temp<count){
        temp++;
        if (temp == count-1) {
            gName[temp]=line;
        }
    }
    if (temp<count-1){
        cout<<"file "<<line<<" overwritten!"<<endl;
    }

    cout<<"file names in array"<<endl;
    for (int i=0;i<count;i++){
        cout<<gName[i]<<endl;
    }

    fil.open("bones.txt", ios::out); //open bones file in output mode
    if (fil.fail()) { //output error message if file fails to open
        cout<<"ERROR! Unable to open bones file!"<<endl;
    } else { //otherwise
        for (int i=0;i<count;i++){ //loop through array line by line
            if (gName[i]!=""){ //if line is not blank
                fil<<gName[i]<<endl; //output array element as line in file
            }
        }
    }
    fil.close(); //close file
    delete [] gName; //clean up memory and delete dynamic array
    cout<<"Saving Game Data!"<<endl;
    fil.open(line+".bin", ios::out | ios::binary); //open binary file in output mode to save game data
    fil.write(reinterpret_cast<char *>(&p1), sizeof(Game)); //write player 1 ship data to file
    fil.write(reinterpret_cast<char *>(&p2), sizeof(Game)); //write player 1 guess data to file
    fil.close(); //close file
}
```

```
//************************   lodGame   ***************************************
//234567890123456789012345678901234567890123456789012345678901234567890123456
789
//000000001111111111222222222233333333334444444444555555555566666666667777777
777
//** Purpose:  //Function to load a saved game
//** Inputs: Game &p1, Game &p2
//** Outputs: none
//****************************************************************************
void lodGame(Game &p1, Game &p2){
   string line=""; //hold file names from bones file. also used as temp to hold names while
counting lines
   fstream fil; //create filestream object
   p1.mapClr(); //reset player 1 to defaults
   p2.mapClr(); //reset player 2 to defaults
   cout<<endl;
   fil.open ("bones.txt", ios::in); //open bones file in input mode
   if (fil.fail()){ //output error if file did not open correctly
      cout<<"ERROR! Unable to open BONES file!"<<endl;
   } else {
      cout<<"List of *Savefiles*: "<<endl;
      while (fil){ //loop through file line by line
         getline(fil, line); //get file line, increment to next line
         cout<<"*"<<line<<"*"<<endl; //output text from that line in file
      }
   }
   fil.close(); //close file

   cout<<"Enter name of game to load: <Do not include * characters>"<<endl;
   cin.ignore(256, '\n');
   getline (cin, line); //enter name of savefile to open
   fil.open(line+".bin", ios::in | ios::binary); //open binary file name provided
   if (fil.fail()){
      cout<<"ERROR! Unable to open file!"<<endl;
      cout<<"Randomly creating new game state!"<<endl;
      putComp(p1); //randomly place player 1 ships
      putComp(p2); //randomly place player 2 ships
   } else { //read in binary file data to populate game array data
      fil.read(reinterpret_cast<char *>(&p1), sizeof(Game)); //read file to fill player 1 ship data
      fil.read(reinterpret_cast<char *>(&p2), sizeof(Game)); //read file to fill player 1 guess data
      fil.close();
   }
   p1.setPlay('C');
   p2.setPlay('C');
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/*
 * File:   AbsGame.h
 * Author: rcc
 *
 * Created on May 24, 2017, 2:42 PM
 */

#ifndef ABSGAME_H
#define ABSGAME_H

class AbsGame{
public:
    virtual char getShip()=0;
    virtual char getGues()=0;
    virtual short getTurn()=0;
    virtual short getHits()=0;
};

#endif /* ABSGAME_H */

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   colors.h
 * Author: scott_r_parker
 *
 * Created on April 10, 2017, 8:57 PM
 */

#ifndef COLORS_H
#define COLORS_H


#define RESET   "\033[0m"
#define BLACK   "\033[30m"      /* Black */
#define RED     "\033[31m"      /* Red */
#define GREEN   "\033[32m"      /* Green */
#define YELLOW  "\033[33m"      /* Yellow */
#define BLUE    "\033[34m"      /* Blue */
#define MAGENTA "\033[35m"      /* Magenta */
#define CYAN    "\033[36m"      /* Cyan */
#define WHITE   "\033[37m"      /* White */
#define BOLDBLACK   "\033[1m\033[30m"      /* Bold Black */
#define BOLDRED     "\033[1m\033[31m"      /* Bold Red */
#define BOLDGREEN   "\033[1m\033[32m"      /* Bold Green */
#define BOLDYELLOW  "\033[1m\033[33m"      /* Bold Yellow */
```

```
#define BOLDBLUE    "\033[1m\033[34m"      /* Bold Blue */
#define BOLDMAGENTA "\033[1m\033[35m"      /* Bold Magenta */
#define BOLDCYAN    "\033[1m\033[36m"      /* Bold Cyan */
#define BOLDWHITE   "\033[1m\033[37m"      /* Bold White */

#define INVERSE     "\033[7m"     /* Swap Bacground and Text colors */
#define UNDERLINE       "\033[4m"     /* Underline Single */

#define BGBLACK     "\033[40m"      /* BLACK Background */
#define BGRED       "\033[41m"      /* RED Background */
#define BGGREEN     "\033[42m"      /* GREEN Background */
#define BGYELLOW    "\033[43m"      /* YELLOW Background */
#define BGBLUE      "\033[44m"      /* BLUE Background */
#define BGMAGENTA   "\033[45m"      /* MAGENTA Background */
#define BGCYAN      "\033[46m"      /* CYAN Background */
#define BGWHITE     "\033[47m"      /* WHITE Background */

#endif /* COLORS_H */

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   val.h
 * Author: scott_r_parker
 *
 * Created on May 22, 2017, 6:57 PM
 */

 /* //input and validate character (char) data.
       //Up to 5 inputs, each input is an acceptable value for the return value
       //of the function...
       char inChar(char inA='\0', char inB='\0', char inC='\0', char inD='\0', char inE='\0')
  *
  *
*/


#ifndef VAL_H
#define VAL_H

#include <iostream>
#include <cctype>
using namespace std;

namespace val {

//input and validate character (char) data.
```

```
//Up to 4 inputs, each input is an acceptable value for the return value
//of the function...
char inAlpha() {
   char choice='\0';
   cin>>choice;
   while (cin.fail() || !isalpha(choice)) {
      cout<<"Enter an alphabetical character only!"<<endl;
      cin.clear();
      cin.ignore(256, '\n');
      cin>>choice;
   }
   return choice;
}
char inAlpha(char low, char high) {
   char choice='\0';
   cin>>choice;
   while (cin.fail() || !isalpha(choice) || (toupper(choice)<low || toupper(choice)>high)) {
      cout<<"Enter a letter from "<<low<<" to "<<high<<"!"<<endl;
      cin.clear();
      cin.ignore(256, '\n');
      cin>>choice;
   }
   return choice;
}

char inChar(char inA='\0', char inB='\0', char inC='\0', char inD='\0', char inE='\0') {
   char choice='\0';
   cin >> choice;
   if (inA=='\0' && inB=='\0' && inC=='\0' && inD=='\0' && inE=='\0') {
      while (cin.fail()) {
         cout<<"Invalid input! Enter an ASCII character!"<<endl;
         cin.clear();
         cin.ignore(256, '\n');
         cin>>choice;
      }
   } else if (inB=='\0' && inC=='\0' && inD=='\0' && inE=='\0'){
      while (cin.fail() || choice!=inA) {
         cout<<"Invalid input! Valid entry is: "<<inA<<endl;
         cin.clear();
         cin.ignore(256, '\n');
         cin>>choice;
      }
   } else if (inC=='\0' && inD=='\0' && inE=='\0'){
      while (cin.fail() || (choice!=inA && choice!=inB)) {
         cout<<"Invalid input! Valid entries are: "<<inA<<", "<<inB<<endl;
         cin.clear();
         cin.ignore(256, '\n');
         cin>>choice;
      }
   } else if (inD=='\0' && inE=='\0'){
      while (cin.fail() || (choice!=inA && choice!=inB && choice!=inC)) {
            cout<<"Invalid input! Valid entries are: "<<inA<<", "<<inB<<", "<<inC<<endl;
            cin.clear();
            cin.ignore(256, '\n');
            cin>>choice;
        }
    } else if (inE=='\0'){
        while (cin.fail() || (choice!=inA && choice!=inB && choice!=inC && choice!=inD)) {
            cout<<"Invalid input! Valid entries are: "<<inA<<", "<<inB<<", "<<inC<<",
"<<inD<<endl;
            cin.clear();
            cin.ignore(256, '\n');
            cin>>choice;
        }
    } else {
        while (cin.fail() || (choice!=inA && choice!=inB && choice!=inC && choice!=inD &&
choice!=inE)) {
            cout<<"Invalid input! Valid entries are: "<<inA<<", "<<inB<<", "<<inC<<", "<<inD<<",
"<<inE<<endl;
            cin.clear();
            cin.ignore(256, '\n');
            cin>>choice;
        }
    }
    return choice;
}

//input and validate a signed int
//first input is minimum value, 2nd input is max value
//Overloaded (no arguments) only validates input TYPE
template <class T>
T inNum(T type, long long int low, long long int high) { //long long int for low/high range
    T x=type; //determines the datatype to use (useful for ranges)
    cin >> x;
    while (cin.fail() || x<low || x>high) {
        cout<<"Invalid input! Enter number from "<<low<<" to "<<high<<"!"<<endl;
        cin.clear();
        cin.ignore(256, '\n');
        cin>>x;
    }
    return x;
}

template <class T>
T inNum(T type) {
    T x=type;
    cin>>x;
    while (cin.fail()) {
        cout<<"Invalid input!"<<endl;
        cin.clear();
        cin.ignore(256, '\n');
        cin>>x;
        }
        return x;
    }

}

#endif /* VAL_H */

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/* 
 * File:   Player.h
 * Author: scott_r_parker
 *
 * Created on April 13, 2017, 2:36 PM
 */

#ifndef MAP_H
#define MAP_H

struct Map{
    short hits;
    short guesses;
    char guess[10][10]={}; //Map to keep track of guesses (hits and misses)
    char ship[10][10]={}; //Map to keep track of player's ships (undamaged, placements, and hit)
};

#endif /* MAP_H */

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/* 
 * File:   AbsGame.h
 * Author: rcc
 *
 * Created on May 24, 2017, 2:42 PM
 */

#ifndef ABSGAME_H
#define ABSGAME_H

class AbsGame{
public:
    virtual char getShip()=0;
    virtual char getGues()=0;
    virtual short getTurn()=0;
    virtual short getHits()=0;
};

#endif /* ABSGAME_H */

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   Game.h
 * Author: Scott Parker
 *
 * Created on May 24, 2017, 2:46 PM
 */

#ifndef GAME_H
#define GAME_H

#include "AbsGame.h"
#include "Map.h"
#include "Guess.h"

class Game; // Forward Declaration

// Function Prototypes for Overloaded Stream Operators
ostream &operator << (ostream &, const Game &);

//All the elements for a game of battleship... 2 maps for each player
// the number of hits and number of total shots fired
class Game {
private:
    Map player; //Player map data
    const int cols=10;
    const int rows=10;
    string lstShot;
    bool playing;
    bool shot;

protected:

public:
    Game();
    virtual ~Game();

    string getLast(); //return lstShot
    void setPlay(char); //set status of playing...
    void mapClr(); //Reset entire game state to empty
    void okShot(Guess); //check to see if shot is hit or miss
    void setMap(Guess, char); //Update map after a valid shot
    void setShip(int, int, int, char); //Place ships on map
    char getShip(int, int) const; //display map data for coordinates
    char getGues(int, int) const; //display map data for coordinates
    short getTurn() const; //Display total number of guesses
    short getHits() const; //Display number of hits
    bool gameOn() const; //return status of playing... if true game in progress, false to exit
    bool getShot() const; //return value of shot

    // Friends
    friend ostream &operator << (ostream &, const Game &);

};

#endif /* GAME_H */

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   Game.cpp
 * Author: Scott Parker
 *
 * Created on May 24, 2017, 2:46 PM
 */

#include "Game.h"
#include "colors.h"

Game::Game() { //initialize game to all ocean spaces
    for (int i=0;i<rows;i++){
        for (int j=0;j<cols;j++){
            player.guess[i][j]='~';
            player.ship[i][j]='~';
        }
        player.guesses=0;
        player.hits=0;
        playing=true;
        shot=false;
        lstShot="No Shots!  ";
    }
}

Game::~Game() {
}
```

```
void Game::mapClr() { //reset to default status
    for (int i=0;i<rows;i++){
        for (int j=0;j<cols;j++){
            player.guess[i][j]='~';
            player.ship[i][j]='~';
        }
        player.guesses=0;
        player.hits=0;
        playing=true;
        shot=false;
        lstShot="No Shots!  ";
    }
}

string Game::getLast() { //return value of lstShot (last shot) text results
    return lstShot;
}

bool Game::gameOn() const { //return value of playing... if false then game ends
    return playing;
}

bool Game::getShot() const { //return value of playing... if false then game ends
    return shot;
}

char Game::getShip(int col, int row) const { //return map value for listed coordinate
    return player.ship[col][row];
}

char Game::getGues(int col, int row) const { //return map value for listed coordinate
    return player.guess[col][row];
}

short Game::getTurn() const { //return the total number of guesses
    return player.guesses;
}

short Game::getHits() const { //return the number of hits
    return player.hits;
}

void Game::setMap(Guess shot, char dat) { //update the map after a valid shot
    if (dat=='o') {
        player.guess[shot.getRow()][shot.getCol()]='M';
        lstShot="MISS!      ";
        player.guesses++;
    } else {
        player.guess[shot.getRow()][shot.getCol()]=dat;
        lstShot="HIT!       ";
        player.guesses++;
        player.hits++;
    }
}

void Game::setPlay(char choice) { //Set the boolean value of playing
    bool retVal=false;
    if (choice=='C') {
        retVal=true;
    } else retVal=false;
    playing=retVal;
}


void Game::okShot(Guess fire) { //Check to see if shot is valid (hit, miss, on map, not duplicate)
    bool tmpShot=false;
    if (player.ship[fire.getRow()][fire.getCol()]=='S') {
        tmpShot=true;
        player.ship[fire.getRow()][fire.getCol()]='H';
    } else if (player.ship[fire.getRow()][fire.getCol()]=='~') {
        tmpShot=true;
        player.ship[fire.getRow()][fire.getCol()]='o';
    } else {
        tmpShot=false;
    }
    shot=tmpShot;
}

//Edit map data to place a ship.
void Game::setShip(int row, int col, int size, char verHor) {
    if (verHor=='v' || verHor=='V') {
        for (int i=0;i<size;i++) {
            player.ship[row+i][col]='S';
        }
    } else {
        for (int i=0;i<size;i++) {
            player.ship[row][col+i]='S';
        }
    }
}

//*******************************************************
// Overloaded << operator. Gives cout the ability to      *
// directly display FeetInches objects.                   *
//*******************************************************

ostream &operator<<(ostream &strm, const Game &obj)
{
    strm << BGBLACK << WHITE << "Legend:  " << BGBLUE << CYAN << "~" << RESET <<
BGBLACK << WHITE
```

```
         << " OCEAN   " << BGRED << YELLOW << "H" << RESET << BGBLACK << WHITE <<
" HIT   "
         << BGWHITE << MAGENTA << "M" << RESET << BGBLACK << WHITE << " MISS
" << endl;
   strm << BGBLACK << UNDERLINE << "         " << WHITE << "S" << " SHIP   " << RESET <<
UNDERLINE << BGWHITE << MAGENTA
         << "o" << RESET << UNDERLINE << BGBLACK << WHITE << " ENEMY GUESS        "
<< RESET << endl;
   strm << BGBLACK << GREEN << "   GUESSES               SHIPS        " << endl;
   strm << BGBLACK << WHITE << "   " << UNDERLINE << "0123456789" << RESET <<
BGBLACK << WHITE
         << "            " << UNDERLINE "0123456789" << RESET << BGBLACK << WHITE << "
" << RESET << endl;
   for (int i=0;i<10;i++) { //display the maps and score data
      strm << BGBLACK << WHITE << static_cast<char>(i+65) << ": ";
      for (int j=0;j<10;j++) {
         if (obj.player.guess[i][j]=='~') {
            strm << BGBLUE << CYAN << obj.player.guess[i][j] << RESET;
         } else if (obj.player.guess[i][j]=='H') {
            strm << BGRED << YELLOW << obj.player.guess[i][j] << RESET;
         } else if (obj.player.guess[i][j]=='M') {
            strm << BGWHITE << MAGENTA << obj.player.guess[i][j] << RESET;
         } else {
            strm << BGBLACK << WHITE << obj.player.guess[i][j] << RESET;
         }
      }
      strm << BGBLACK << WHITE << "          " << BGBLACK << WHITE <<
static_cast<char>(i+65) << ": ";
      for (int j=0;j<10;j++) {
         if (obj.player.ship[i][j]=='~') {
            strm << BGBLUE << CYAN << obj.player.ship[i][j] << RESET;
         } else if (obj.player.ship[i][j]=='H') {
            strm << BGRED << YELLOW << obj.player.ship[i][j] << RESET;
         } else if (obj.player.ship[i][j]=='o') {
            strm << BGWHITE << MAGENTA << obj.player.ship[i][j] << RESET;
         } else {
            strm << BGBLACK << WHITE << obj.player.ship[i][j] << RESET;
         }
      }
      strm  << BGBLACK << "   "<< '\n';
   }
   strm << BGBLACK << WHITE << " HITS: " << obj.player.hits << "   " << "SHOTS: "
         << obj.player.guesses << "   " << "Last: " << obj.lstShot << RESET;
   return strm;
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/*
 * File:   Guess.h
 * Author: scott_r_parker
 *
 * Created on May 24, 2017, 4:54 PM
 */

#ifndef GUESS_H
#define GUESS_H

#include <iostream>
using namespace std;

class Guess; // Forward Declaration

// Function Prototypes for Overloaded Stream Operators
ostream &operator << (ostream &, const Guess &);
istream &operator >> (istream &, Guess &);

class Guess{
private:
   int targRow=11;
   int targCol=11;

public:
   Guess();

   void setRow(int); //set the row value manually
   void setCol(int); //set the column value manually
   int getRow(); //return the row guess
   int getCol(); //Return the column guess

   // Friends
   friend ostream &operator << (ostream &, const Guess &);
   friend istream &operator >> (istream &, Guess &);

};

#endif /* GUESS_H */


/*
 * File:   Guess.cpp
 * Author: Scott Parker
 *
 * Created on May 24, 2017, 2:46 PM
 */


#include <cctype>
```

```
#include "Guess.h"

Guess::Guess() {
   targRow=-1;
   targCol=-1;
}

void Guess::setRow(int a) {
   targRow=a;
}

void Guess::setCol(int a) {
   targCol=a;
}

int Guess::getCol() {
   return targCol;
}

int Guess::getRow() {
   return targRow;
}

//*******************************************************
// Overloaded << operator. Gives cout the ability to     *
// directly display FeetInches objects.                *
//*******************************************************

ostream &operator<<(ostream &strm, const Guess &obj)
{
  strm << obj.targRow << ", " << obj.targCol;
  return strm;
}

//*******************************************************
// Overloaded >> operator. Gives cin the ability to      *
// store user input directly into FeetInches objects.    *
//*******************************************************

istream &operator >> (istream &strm, Guess &obj)
{
   char temp='\0';
   // Prompt the user for the vertical axis (row)
   cout << "Enter target Row (A-J): ";
   strm >> temp;
   temp=toupper(temp);
   while (cin.fail() || !isalpha(temp) || (temp<65 || temp>74)) {
      cout<<"Enter target Row A-J only! ";
      cin.clear();
      cin.ignore(256, '\n');
      strm >> temp;
```

```
        temp=toupper(temp);
    }
    int conTemp=0;
    conTemp=temp-65; //convert character to int
    obj.targRow=conTemp;

    // Prompt the user for the horizontal axis (column)
    cout << "Enter target Column (0-9): ";
    strm >> obj.targCol;
    while (cin.fail() || obj.targCol<0 || obj.targCol>9) {
        cout<<"Enter target Column 0-9 only! ";
        cin.clear();
        cin.ignore(256, '\n');
        strm >> obj.targCol;
    }

   return strm;
}
```