3460:209 Programming Assignment 4 Sharks and Battleships ⎈

Well you won the popular vote to be captain but lost in the Electoral College. Tough break. Your first task as first mate is to guide the ship out of harbor and onto the open sea. You will have to avoid numerous perils along the way.

# Overview

The area between the harbor and the hole in the reef can be represented by a 15-by-30 grid where each point representing a square nautical mile. Contained in this grid are:

* One British battleship on patrol.
* Very hungry great white sharks that fancy pirates (they taste like chicken marinated in Guinness).
* Random icebergs. In addition, the entire perimeter of the oceanic area described is made up of icebergs that formed overnight.
* A single exit point from the wall of surrounding icebergs that will release your ship into the open sea for escape.

Your program will repeatedly ask you which direction you want to move in. Each time you move, you must try to avoid hitting, or being hit/eaten by an iceberg or a shark (remember, they’re very big and like to eat pirates), or being blasted by the Brits. You are within range of the Brits if you are in an adjacent square to their war vessel, including diagonal squares. It’s almost certain that your ship will be obliterated by cannon fire.

# Function Decomposition

You are being provided with the functions that you will need below. You simply need to add these to your program and complete the code and commit your program to SVN (see submission). The function prototypes and purposes are as follows:

* void briefing();

Give the user playing instructions.

* void readMap(char ocean[ROWS][COLUMNS], int, int, int, int);

Read in the initial map from a file (this is the map.txt file). Parameters include the map, your current position and the coordinates of the hole in the wall. The file contains exactly 15 lines of 30 integers per line, coded as follows:

* + 0 = open sea
  + 1 = iceberg
  + 2 = shark
  + 3 = British battleship
  + 4 = the escape position
  + 5 = your pirate ship

When you create the actual map you are to insert/convert to the actual characters to be displayed into the array.

* + Your ship and mates are represented by the character ‘®’. (Arrgh!)
  + The battle ship is represented by an asterisk ‘\*’.
  + Sharks are represented by the Latin symbol ‘œ’ (which *sort of* looks like a fish with its mouth open).
  + Icebergs are represented by a solid dot ‘•’.
  + Open sea is represented by a blank ‘ ’.
  + The escape position is represented by an ‘X’.

You read in the map as a 2-dimensional array of integers for the sole purpose of file reading sanity. However, this is not how it will appear when you display it on the screen. You may not use the integer array to represent your ocean. As you read numbers, populate a 2-dimensional character array appropriately and proceed when finished reading the file.

* void printMap(const char ocean[ROWS][COLUMNS]);

Print the current ocean grid map. The ocean map should be displayed on the screen after each time you move and after the battleship and all of the sharks have moved.

* bool pirateMove(char ocean[ROWS][COLUMNS], int myX, int myY, const int escX, const int escY);

Get a move from the player. Remember, each time you move all the sharks and the battleship move one space toward you. Parameters are the same as for readMap(). Keyboard commands from the numeric keypad will move your ship as follows:

|  |  |  |
| --- | --- | --- |
| ❼ | ❽  move one row above  your current position | ❾ |
| ❹  move one column to the left  of your current position | ❺ | ❻  move one column to the right  of your current position |
| ❶ | ❷  move one row below  your current position | ❸ |

Any other keyboard entry will keep your ship in the same position. (Optionally you can implement diagonal moves for the ship if you choose.) After moving the ship, check to see if your voyage is to continue (see endVoyage() below). If so return true, otherwise false.

* void endVoyage(int outcome);

Check to see if you have reached the escape point. There are four possibilities:

1. You’ve found the hole in the wall and escaped.
2. You’ve run into an iceberg.
3. You’ve been eaten by the sharks.
4. You’ve been destroyed by the British patrols.

Accept as an argument one of these options and print an appropriate message explaining what happened.

* bool sharkMoves(char ocean[ROWS][COLUMNS], int myX, int myY);

Move the sharks. Each time you move your ship, the battleship and sharks move toward you (see below). Unlike your ship sharks can move diagonally (northeast, northwest, southeast and southwest). Sharks cannot travel through icebergs and therefore avoid them. The shark must still move toward you but it may not be optimal. Two sharks may not hit each other. If two sharks collide one is left.

* bool battleshipMove(char ocean[ROWS][COLUMNS], int myX, int myY);

Move the battleship. The battleship can also navigate diagonally (northeast, northwest, southeast and southwest) and destroys anything in its path – icebergs, sharks, and, unfortunately, you. That means if a battleship moves to a position where an iceberg was, when it moves away, that map position becomes open sea.

The game continues until you reach the exit point; or until the sharks or the Brits have eliminated you; or you carelessly navigate into an iceberg. Once the scenario has reached a conclusion, print out what happened.

# Further Coding Details

* Make sure you use call by value and call by reference properly. Don’t automatically assume that the function examples given have headers and prototypes that are correct.
* Use a two-dimensional character array to represent the oceanic grid map.
* At a minimum you need functions previously described
  + Give the user playing instructions.
  + Read in the initial map.
  + Print the current ocean grid map.
  + Get a move from the pirate (that’s you by the way) and move to that position.
  + Move the sharks and battleship.
  + Check to see if you have reached the escape point, been blown up or eaten.
* Functions must have header comments that include the purpose of the function. They also must include the pre-condition and post-condition documentation as well.
* Use appropriate code comments, variable names, and constants.
* Use good coding practices (i.e., spacing, indentation, etc.)

# Submission Details

* You will need to create a program file called main.cpp in the Projects/Project4 folder in SVN.
* You will use the map.txt file for your program’s input. This file is in the Projects/Project4 folder.
* Commit your final main.cpp to SVN. On the student Subversion repository, navigate to Projects/Project4. Create programs formatted under the name of the project.
* Remember that it’s a *good* thing to commit as you develop. This is an excellent back up for you. I don’t look at the number of commits you make. I will only grade the last commit after the due date.
* **You ARE permitted to work with your peer**. Consult with them, me, the graders or the tutors for help, but be prepared with actual code to work with.
* **We will not grade a commit made after 11:59 p.m. on Friday, May 5, 2015.**

*Last updated 11.15.2014. Based on material by K. Liszka.*