CS 132

Project 1: "Sink the Fleet"

Names:

You have already been to "boot camp" and learned the basics of "C". Now it's time to head into battle and use this knowledge, along with a few new "C++" tools.

Your Objective: Sink the Fleet.

To create this project, you will be using a form of extreme programming. In order to do this, you will need to find a single partner and form a programming pair. You will turn in one project per team.

The premise of the game is that two opposing fleet commanders are caught in a thick fog. Both commanders know that the enemy fleet is nearby, but neither knows the exact positions of the ships. Following the chivalrous rules of sea battle, the two commanders take turns firing at specified coordinates. In the case of a "hit," a commander may continue firing until there is a miss. The battle continues until all of one fleet's ships are sunk.

You will need to write this game following the given specifications.

In order to do this, you will need to create some new data types:

The *enumerated* type ***Direction*** should be defined to have the following possible values:
HORIZONTAL, VERTICAL

The *enumerated* type ***Ship*** should be defined to have the following possible values
NOSHIP, MINESWEEPER, SUB, FRIGATE, BATTLESHIP, CARRIER, HIT, MISSED

A **Cell** should be defined as a **struct** which contains:

- m_row (type unsigned short), the row of the cell
- m_col (type unsigned short), the column of the cell

A **ShipInfo** should be defined as a **struct** which contains:

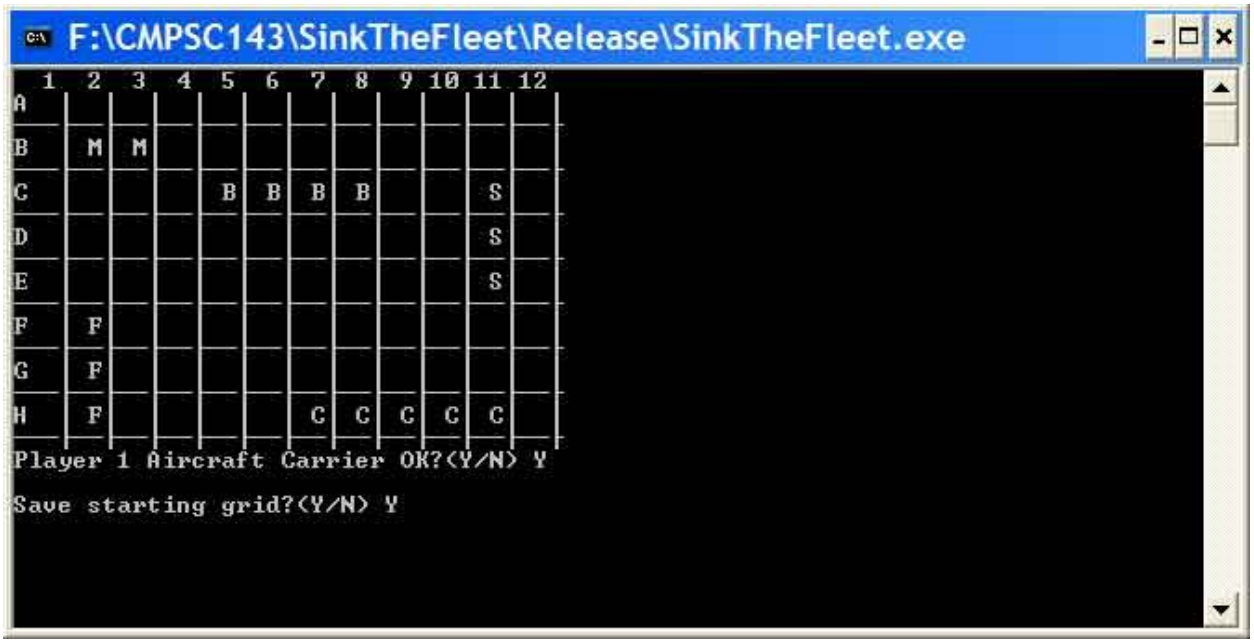
- m_name (type Ship), which ship?
- m_orientation (type Direction), which direction is the ship facing?
- m_bowLocation (type Cell), which cell is the bow location?
- m_piecesLeft (type short), how many sections of the ship are left undestroyed?

A **Player** should be defined as a **struct** which contains:

- m_gameGrid (an array of 2 Ship**), an array of 2 dynamically allocated 2-dimensional arrays of Ships.
  (each game grid has an array of two 2D arrays -- [0] is the player's own ships, [1] is a record of shots fired--marking hits and misses)
- m_ships (an array of 6 ShipInfo), the ships in the fleet
- m_piecesLeft (type short), how many sections of the ships in the fleet are left undestroyed?

Your program should work in the same manner as the sample. It must allow the user to choose a game size (S or L), and should dynamically allocate the appropriate size of game grids.

The program must allow each user to enter the initial grid from either the keyboard or from a file. The program should not accept improper input, bad filenames or bad files, ships extending out-of-bounds or overlapping ships. If directly inputting ships, the user should be given the option of saving the grid to a file-- which should just be a printout of "S" or "L" on a separate line, the location and orientation of each ship, one per line followed by a printing of the grid (*using the same function that prints the grid to the screen*). *--Run the sample program and save a grid to see a sample file.*



Deliverables:

Physical:

The Project should be turned in inside a clear **plastic** file folder. This folder should have a simple flap to hold paper in place--NO buttons, strings, Velcro, etc. Pages should be in order, not stapled.

•Assignment Sheet *(printed pdf from the web),* with your **name** written on it, as a cover sheet.

•Printed **Source Code** with Comments (*including heading blocks -- a file header for each file plus a function header for **each** function. Describe parameters, any input or output, etc., no line wrapping). Print in portrait mode, 10 - 12 point font.*

Electronic:

• All ***.h***, ***.cpp, .exe****(Release Version)*, and saved output (***.shp***) files, ***zipped*** together. **Do not** use ***rar*** or any archive format other than ***zip***. Rename the file: "<*YourNames*>_p1.zip".

•**Sample** Output *(as .rtf -- run the program, copy the window using <ALT|PRTSCN>, paste into Paint, invert colors (<Ctrl|Shift|I>), copy, open Wordpad, save.)*

• a simple **test plan** including explanations of any discrepancies and reasons for each test. Show actual input and **ALL** values output as well as **ALL** expected output. Test each possible **action**. Save as.*xls, xlsx, .doc* or *.docx* file

•Submit this single ***zip*** file by going to **canvas**, select this class, select the Assignment tab on the left, select Assignment 1, select the submission tab at the top right, find the file, and Submit.

Due:

Section A: Tuesday, February 2, 2016, 9:30 a.m. *(beginning of class)*
Section B: Tuesday, February 2, 2016, 2:50 p.m. *(beginning of class)*
Resubmittal due: Two weeks from the day the project is returned in class (*beginning of class*).