**Project 1: Build a system - Team’s choice of language and platform**

**Due Date: 11:59 PM, Sunday, February 21, 2021**

* All teams must have a code freeze as of 11:59 PM, Sunday, February 21, 2021.
* The timestamp will be judged by the final commit on your master branch of the team's Github repository.
* You can continue to work on other branches but cannot update your master branch after the freeze date.
* You must demo your code the week of February 22 in your lab based on that master branch as of the code freeze.
* All artifacts (code and documentation) must be in your repository on your master branch.
* Your CATME Peer Reviews for each team member are also due at this time.

**Overview**

* Let's make Battleship!
* Battleship is a two player game.
* Both players secretly place 1 to 6 ships on a 10x10 grid.
* Taking turns, each player announces where on the opponent's grid they wish to fire.
* The opponent must announce whether or not one of the ships was hit.
* The first player to sink all of the opponent's ships wins.

**Requirements**

1. Game Setup
   1. Board size
      1. 10x10
      2. The columns are denoted by letters (A-J)
      3. The rows are denoted by number (1-10)
   2. Number of ships (per player)
      1. Given by user
      2. Minimum of 1 and a maximum of 6
   3. Types of ships
      1. This will be based on the amount of ships chosen.
      2. If a total of 1 ship is chosen, then each player gets a single 1x1 ship
      3. If a total of 2 ships is chosen, then each player gets a 1x1 and a 1x2 ship
      4. This continues up to 6, where each player will a 1x1, 1x2, 1x3, 1x4, and a 1x6 ship
   4. Ship placement
      1. After the amount of ships is chosen, players need to be able to secretly orient and place the ships on their board.
2. Playing the Game
   1. Taking turns, the users pick a space on the opponent's board to "fire" at.
   2. They must then be informed if the shot was a "hit" or a "miss".
   3. The player's view should be updated to reflect this (see Player's view below).
   4. After each shot, it is the other player's turn.
3. Destroying a ship
   1. Once a ship has been hit in every space it occupies, it is sunk.
   2. For example, if the 1x3 ship occupies the space B3, B4, and B5.
   3. Once the opponent has shot those three spaces, that ship is sunk.
4. Player's view
   1. A player should have full view of their board and where their ships are placed.
   2. Show how many times each ship has been hit
   3. A player should have a board to track all shots they've fired and whether they were misses or hits.
5. Game End
   1. Once a player has sunk all of the opponent's ships, they immediately win.

**Language and Platform**

* Team’s choice

**Plagiarism**

* You can use existing libraries, but you must cite all sources of code you did not author in your documentation.
* This assignment, or variations of it, has been done in other EECS 448 classes.
* **You may not use code from those classes (even if you cite it)!**
* Like any good programmer would do, you may certainly look at these projects for ideas, but you may not use any of the code.

**Grading Rubric**

* Team Score (80% of grade – team based)
* This portion of the project will be graded by your GTA.
* The project points are broken down into the following sections.
* Demo (40%)
  + You will demo on a device of your choice in the lab during the week of February 21.
  + Specified features present
  + Stability - GTA will stress test your application; crashes, memory leaks, or other things that you also hate in bad software will be met with a penalty
  + User interface - your product should be intuitive to use; good rule to stick by: If I need a manual to use your interface, you have a bad interface.
  + Modularity - your code should be easily extensible and divided into logical components (i.e. not one class that does everything)
* System Documentation (40%)
  + On your github repository, have a folder called "documentation" that contains all the generated documentation, readmes, work cited list, and any other needed documentation.
  + Estimate of person-hours for completing the project (this should be done as soon as possible – 10%)
    - Provide the details of how you arrived at the estimate
  + Actual accounting of the person-hours required to complete the project (10%)
    - This needs to be a day-by-day accounting from each team member on how many hours they spent on the project, including team and GTA meetings, coding, testing, documenting, etc. Do not include time attending 448 lectures or working on in-class problems.
    - You WILL NOT be penalized for taking longer (or shorter) than your estimate.
    - You WILL be penalized, if you do not have a day-to-day accounting or it looks like to the GTA that you created the accounting just to meet this requirement.
  + You must use some kind of documentation format/software, such as JavaDoc from EECS 268, to provide documentation of your code (10%)
    - In short, you need to use some means of documentation that can be used to generate HTML documentation of your code.
    - Your GTA should be able to open an HTML file to find out all the class information such as method names, signatures, pre and post conditions, etc.
  + Works cited section (10%)
    - This will be its own section in the documentation.
    - Any code that you or a teammate did not write from scratch, must be cited.
* Team Evaluations (20% of grade – individual based)
* This part will be determined by CATME Peer Review (which will be explained in Lab 3).