EGME 205 Group Project

Spring 2020: Battleship

Due: Monday, May 4, 2020

Your goal is to program a Matlab version of "Battleship." In this two-player game, each player starts with five ships, each of which has a certain length (see Table 1 below).

Table 1: Ships.

Ship	**Length**
Carrier	5
Battleship	4
Destroyer	3
Submarine	3
Patrol Boat	2

At the beginning of the game, each player arranges their ships on a 10×10 grid (called that player's *primary grid*). Players then take turns guessing points on their opponent's grid where they think the opponent's ships are. If a guessed location contains part of a ship, that ship gets "hit." When all points on a ship have been hit, that ship is "sunk." Each player keeps track of both players' guesses, hits, and sunken ships on their primary and secondary grid, the latter of which represents their opponent's grid (see Figure 1). The first player to sink all of the other player's ships wins.

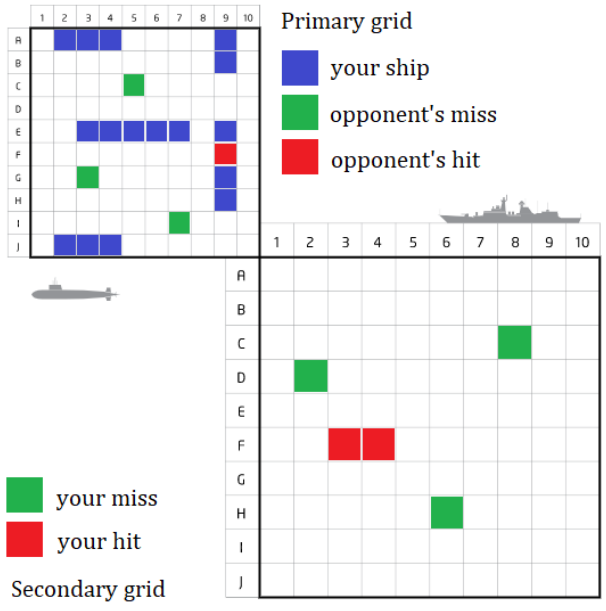

Figure 1: Primary and secondary grids.

(a) **(40 points) Two-Player Mode** Start by programming a two-player version of the game in which two human players play against each other. Once the two players have arranged their ships, the game should only display the *secondary grids* for each player—otherwise, the players would know where their opponent's ships were.

(b) **(40 points) One-Player Mode** Once you have finished the two-player mode, program a one-player version in which a single human player plays against an artificially intelligent computer player. The one-player mode should have two different difficulties: Easy and Hard. In Easy mode, the computer player should arrange their ships randomly and guess randomly. In Hard mode, the computer should have some kind of strategy for arranging their ships and for guessing. You are encouraged to research the best strategies for Battleship when programming the computer player. The one-player version should display the player's primary and secondary grids.

The first thing your code should do is prompt the user to select one-player mode or two-player mode. If the user selects one-player mode, the code should then prompt the user to select the difficulty. Throughout the course of the game, when prompting the player to do something, the game must not allow the user to select anything other than valid options.

(c) **(20 points) Enhancements** In addition to the requirements listed above, you must include at least three enhancements to the game, unique to your group. These enhancements should add an element of originality to your code, separating it from the rest of the groups. Ideally, these will make the game more enjoyable for the player. To receive credit for your enhancements, you must describe them in a comment at the beginning of the code.

All code must be your own, and you may only use the following commands and structures that were covered in this course:

- arrays/matrices
- basic math
- plot() and related functions
- Boolean logic and logical operators
- isequal()
- IF-statements
- WHILE-loops
- FOR-loops
- characters and strings
- input()
- fprintf()
- rand
- round(), ceil(), floor()
- mod()
- user-defined functions

Use of any other commands or structures will result in a zero grade for the project. If one of your enhancements requires a command or structure that is not listed above, you may

request an exception for your group. All requests must be made to the instructor in writing via email at least one week prior to the due date. The instructor reserves the right to refuse such a request for any reason.

Academic dishonesty of any kind, as defined in the course syllabus, will result in an automatic grade of F for the course and referral to the Office of Student Conduct for disciplinary action. If at any time you have a doubt as to whether what you are doing is acceptable, please seek clarification from the instructor immediately. It is always better to err on the side of safety.

Your final submission must be a single M-file named "battleship_groupX.m," where "X" is the letter corresponding to your assigned group. Your code will be graded according to the rubric in Table 2.

Table 2: Grading rubric.

Quality	**Score**	**Description**
High	100%	Meets all of the requirements; no bugs/errors; code runs as intended; three or more highly original enhancements
Medium	80%	Meets all of the requirements; no bugs/errors; code runs as intended; fewer than three enhancements or three or more unoriginal enhancements
Low	50%	Meets some, but not all, of the requirements; at least one bug/error that prevents the code from running as intended
Very Low	25%	Does not meet the requirements, but an honest attempt was made
Poor	0%	Does not meet the requirements; very little, if any, attempt was made