Steph McCallum & Nathan Smith

CST 221 & CST 235

Course Code: COSC295

Instructor Name: Sharon McDonald

Document Name: Battleship – Acceptance Tests.docx

Document Purpose: Outline testing done on Command- Line Battleship game build in Swift.

Assignment Number: 2

Due Date: 2017Apr06

Acceptance Testing: Command Line Battleship

# COSC295 Assignment #2

ASSUMPTIONS

* The following Swift files will be downloaded and compiled (using either ‘swift build’ or ‘swiftc <files>’):

board.swift boat.swift main.swift

* A game MUST have 5 boats total
* it is acceptable that out of the three types there is a chance that none are generated from one of the types (i.e. /1 sub, 4 carriers, 0 tugboats).
* We have a cheat code, and a help code.

TESTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST | PROGRAM INPUT | EXPECTED RESULT | ACTUAL RESULT | PASS/FAIL |
| User can start a new game. | User runs the command ‘./main’ | User is shown the welcome header, and a new board. |  | PASS |
| User cannot see where generated boats are when the game begins. | None | All spaces that can be fired on, are displayed as “\*” |  | PASS |
| User can select a valid letter and number to fire on a tile with. Tile is empty, so nothing is displayed. | User enters the following:  **Letter:** A **Number:** 0 | A blank space is shown instead of “\*” or a boat. |  | PASS |
| User can select a valid letter and number to fire on a tile with. Tile is NOT empty, so boat symbol is displayed. | User enters the following:  **Letter:** C **Number:** 2 | Tugboat symbol is displayed. |  | PASS |
| User cannot select an invalid letter to fire on. | User enters the following: **Letter:** Z | An error message is displayed to the user. User is prompted again for input. |  | PASS |
| User cannot select an invalid number to fire on. | User enters the following:  **Letter:** C **Number:** 65 | An error message is displayed to the user. User is prompted again for input. |  | PASS  (Prompt restarts from the beginning due to error checks) |
| User can invoke the help method and learn of the number of ships generated in this instance of the game. | User enters the following:  **Letter:** ? | A message is displayed to the user, indicating the number of submarines, carriers, and tug boats currently in the game. |  | PASS |
| User cannot invoke the help method from the ‘number’ input. | User enters the following: **Letter:** A **Number:** ? | An error message is displayed to the user. User is prompted again for input. |  | PASS (This simplifies our error checking on the number input, and the user is still able to invoke ? when they are re-prompted, so this functions exactly as intended.) |
| User can ‘cheat’ and wipe the board. | User enters the following: **Letter:** ! | All asterisk are removed from the board. All boat locations are shown. Game is treated as over; user is asked if they would like to play again. |  | PASS (You didn’t ask for a cheat, but we implemented one to help with debugging.) |
| User cannot enter symbols (other than ‘?’ or ‘!’) to fire upon. | User enters the following:  **Letter:** @ | An error message is displayed to the user, indicating that they can only enter letters A through H, or ‘?’ for help. User is prompted again for input. |  | PASS |
| User cannot enter symbols to fire upon, when prompted for a number. | User enters the following:  **Letter:** A **Number:** @ | An error message is displayed to the user, indicating that they can only enter numbers 0 through 9. User is prompted again for input. |  | PASS |
| User cannot enter a string of letters to fire upon. | User enters the following: **Letter:** AAAAAAAAA | An error message is displayed to the user, asking them to enter only a single character for input. User is prompted again for input. |  | PASS |
| User is notified that they sunk a ship, when they have hit the ship for however many tiles it is long. | User has three Ш displayed on the board at B5, B6, B7. User enters the following:  **Letter:** B  **Number:** 8 | The final Ш tile is displayed. A message is displayed to the user indicating that they sunk a carrier. User is prompted again for input. |  | PASS |
| User is asked if they want to play again, once all boats have been revealed on the board. | All boats have been revealed on the board (whether by the sinking them, or by cheating) | User is asked if they would like to play again. |  | PASS |
| User can enter ‘y’ to play the game again. | All boats have been revealed on the board (whether by the sinking them, or by cheating). When prompted, user enters ‘y’. | A new, covered board is displayed. User is prompted for where to fire. |  | PASS |
| User can enter anything other than ‘y’ or ‘Y’ to exit the game. | All boats have been revealed on the board (whether by the sinking them, or by cheating). When prompted, user enters a number, symbol, or literally anything other than ‘y’ or ‘Y’. | A goodbye message is displayed. Game ends, command prompt is displayed again. |  | PASS |
| Games create a random assortment of submarines, carriers, and tugboats (five boats in each game). | User invokes ‘?’ in game to see how many of each boat currently exist in game #1. User starts a new game. User invokes ‘?’ to see how many of each boat exist in game #2. | Game #1 has a different assortment of boats than Game #2. | Game #1:    Game #2: | PASS |
| Game randomly places boats on the board; boats can be vertical or horizontal. | User invokes ‘!’ to blow away the board and see the layout of two different games. | Layout of Game #1 is different than Game #2. (Note: not the same games as the above test) | Game #1:    Game 2: | PASS (Bug: nothing will be placed in Row J, nor column 9) |
| User is displayed an error if they hit “enter” (newline) when prompted for a response. | User is prompted for any response, at any point of the game. | User is displayed an error message indicating that the input was invalid. |  | PASS |