*Owen Lindsey*

*Professor Demland, David*

*CST-201 Battleship*

*10/20/2024*

**Pseudo code for battleship**

*Pseudo code solution for PlaceSubmarine:*

***METHOD*** *PlaceSubmarine(board, row, column, submarine)*

***SET*** *offsets to: (0,0), (1,1), (2,2)*

***IF*** *submarine is upwards-oriented* ***THEN***

***FOR EACH*** *offset in offsets*

***NEGATE*** *offset's row value*

***END FOR***

***END IF***

***IF*** *submarine is left-oriented* ***THEN***

***FOR EACH*** *offset in offsets*

***NEGATE*** *offset's column value*

***END FOR***

***END IF***

*FOR EACH offset in offsets*

***SET*** *newRow to row + offset.Row*

***SET*** *newColumn to column + offset.Column*

**Pseudo code for battleship**

*Pseudo code solution for PlaceSubmarine:*

***IF*** *newRow is out of bounds OR newColumn is out of bounds* ***THEN***

*RETURN false*

*END IF*

***IF*** *board[newRow, newColumn] is not empty* ***THEN***

***RETURN*** *false*

***END IF***

***END FOR***

***FOR EACH*** *offset in offsets*

***SET*** *newRow to row + offset.Row*

***SET*** *newColumn to column + offset.Column*

***SET*** *board[newRow, newColumn] to occupied*

***END FOR***

***RETURN*** *true*

***END METHOD***

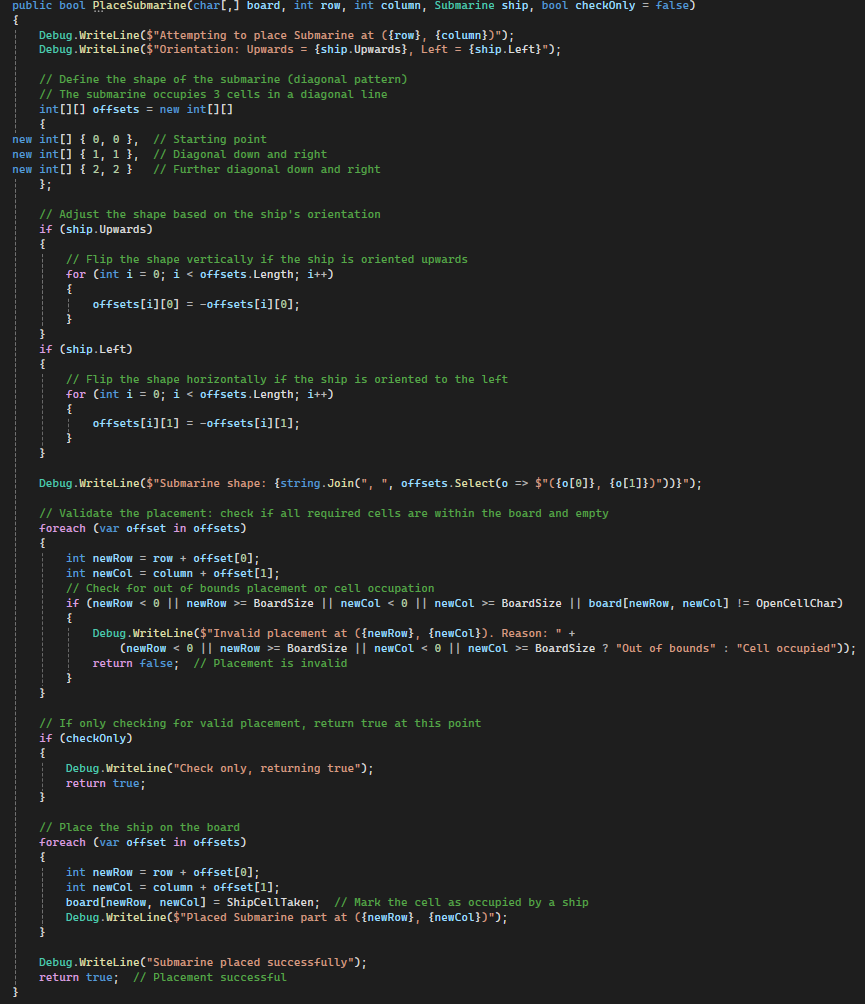
**Test Case for PlaceSubmarine:**  
*Summary of PlaceSubmarine Test Requirements:*

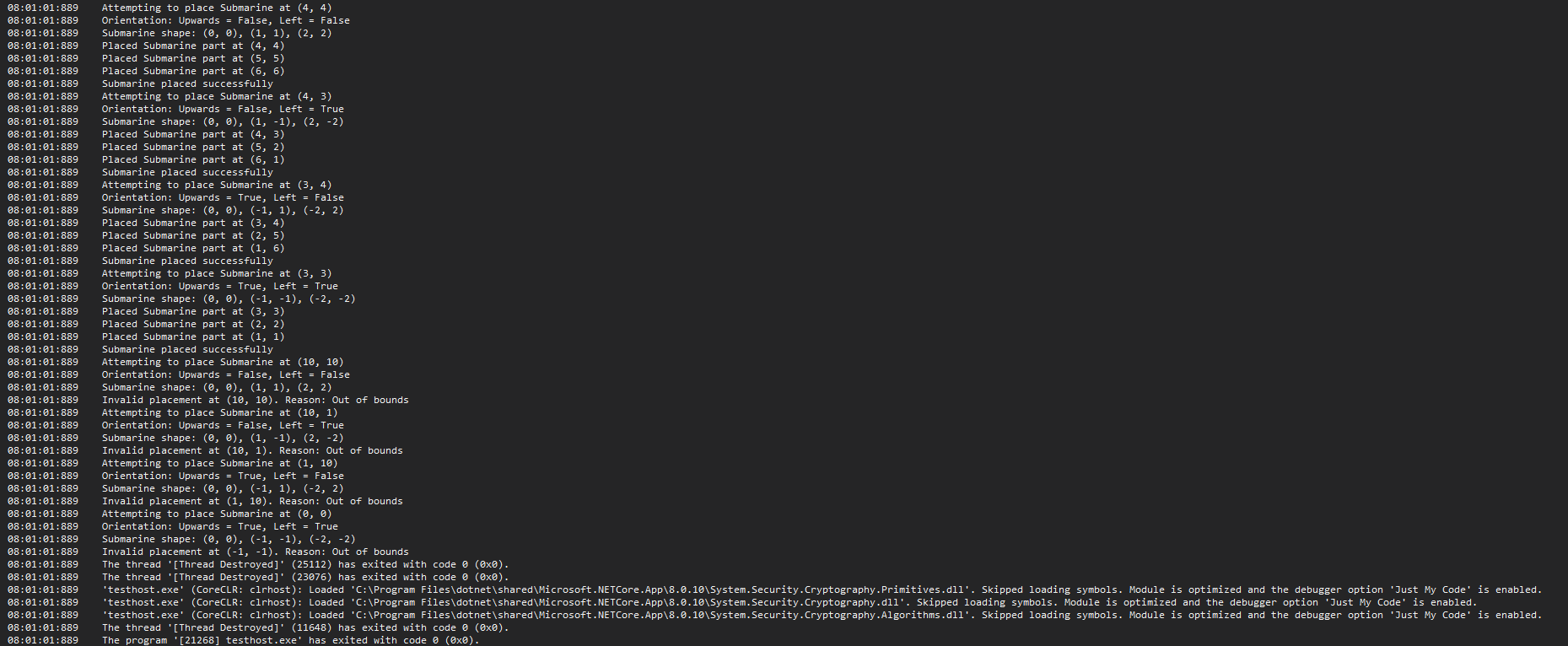
*1.*Standard Placement (down-right diagonal):

* 1. Place at (4,4), occupying (4,4), (5,5), (6,6)

1. Left-oriented Placement (down-left diagonal):
   1. Place at (4,3), occupying (4,3), (5,2), (6,1)
2. Upward Placement (up-right diagonal):
   1. Place at (3,4), occupying (3,4), (2,5), (1,6)
3. Upward and Left-oriented Placement (up-left diagonal):
   1. Place at (3,3), occupying (3,3), (2,2), (1,1)
4. Invalid Placements:
   1. Out of bounds: Attempt to place at (10,10) standard orientation
   2. Out of bounds: Attempt to place at (10,1) left-oriented
   3. Out of bounds: Attempt to place at (1,10) upward-oriented
   4. Out of bounds: Attempt to place at (0,0) upward and left-oriented
5. Why PlaceSubmarine Passes All Test Cases:
6. Flexible Orientation:
   1. The method correctly handles all four diagonal orientations by adjusting the offsets based on the submarine's properties (Upwards, Left).
7. Diagonal Placement:
   1. It correctly implements the unique diagonal shape of the submarine, occupying three cells in a diagonal line.
8. Boundary Checking:
   1. Before placing the ship, it checks if all required cells are within the board boundaries. This ensures it catches all out-of-bounds placement attempts.
9. Collision Detection:
   1. It verifies that all required cells are empty (OpenCellChar) before placement, preventing overlap with existing ships.
10. Atomic Placement:
    1. The method only places the ship if all checks pass. If any check fails, it returns false without modifying the board.

**Test Case for PlaceSubmarine:**  
*Screenshot of method PlaceSubmarine:*



**Test Case for PlaceSubmarine:**  
*Screenshot of Debug output of PlaceSubmarine test case:*  


**Test Case for PlaceSubmarine:**  
*Screenshot of success output of PlaceSubmarine test case:*  
