CECS 277 – Lab 3 – 2D Lists

Battleship

Create a program that allows the user to input coordinates onto a 5x5 2D list in order to locate and destroy a hidden enemy ship. The game begins with a blank board similar to the one below:

```
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
```

Randomly generate the location of a 2x2 enemy ship but do not display it on the board (Hint: either create a solution board with the ship on it, or create a 1D two-item list that stores the row and column of the upper left corner of the ship, and then use that location to calculate the rest of the ship).

The user fires a shot by specifying the row (A-E) and then the column (1-5). Check to make sure that the input is a valid entry (ie. within the bounds of the list and is not a location that has already been chosen). If the shot hits the ship then place a '*' symbol in that location to show a 'hit'. Otherwise, place an 'x' in that location to show a 'miss'.

When the user has hit all four spots that the ship occupies then the user has won. Reset the board and re-randomize the ship so that the user can play again. Give the user the option to give up, which reveals the location of the enemy ship. Once the user has seen the solution, the game should be reset again.

Create the following functions in your program:

1. `display_board(board)` – passes in the 5x5 game board, display the contents of the board with row and column headings similar to the above.
2. `reset_game()` – reset the game board and randomize the position of ship. Return the game board and the solution.
3. `get_row()` – prompt the user to enter a row A-E. Validate the user input, repeatedly reprompt if it is invalid, then and return the corresponding integer location 0-4.
4. `fire_shot(board, solution, row, col)` – passes in the 2D game board, the solution, and the user's row and column input. Using the user's input, and the solution, place an 'x' on the grid if the location was a miss, or a '*' if it was a hit. Return True if it was a hit, and False otherwise.

The main function should call reset_game to initialize the board and generate the solution. Display the main menu and get the user's input. If they choose to play the game, then get the user's row and column input. If it is a valid input, then call fire_shot to determine if their guess was a hit or a miss. If they win or view the solution, then reset the board again. Repeat until they choose to quit the game.

Example Output:

```
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
```

```
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): b
Enter a Column Number (1-5): 2
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ x ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): b
Enter a Column Number (1-5): 4
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ x ~ * ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): b
Enter a Column Number (1-5): 4
You have already chosen that
location.
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ x ~ * ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): b
Enter a Column Number (1-5): 5
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ x ~ * *
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
x
Invalid input - should be an
integer.
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): a
Enter a Column Number (1-5): 4
  1 2 3 4 5
A ~ ~ ~ x ~
B ~ x ~ * *
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): c
Enter a Column Number (1-5): 4
  1 2 3 4 5
A ~ ~ ~ x ~
B ~ x ~ * *
C ~ ~ ~ * ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): c
Enter a Column Number (1-5): 5
You won!
  1 2 3 4 5
A ~ ~ ~ x ~
B ~ x ~ * *
C ~ ~ ~ * *
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~

  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): b
```

```
Enter a Column Number (1-5): 3
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ * ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
2
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ * * ~ ~
C ~ * * ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
2
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ * * ~
E ~ ~ * * ~
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ ~ ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
1
Enter a Row Letter (A-E): d
Enter a Column Number (1-5): 3
  1 2 3 4 5
A ~ ~ ~ ~ ~
B ~ ~ ~ ~ ~
C ~ ~ ~ ~ ~
D ~ ~ x ~ ~
E ~ ~ ~ ~ ~
Menu:
1. Fire Shot
2. Show Solution
3. Quit
3
```

Notes:

1. Place your name, the date, and a brief description of the program in a comment block at the top of your program.
2. Use the check_input module provided on Canvas to check the user's input for the main menu and column input (not row input).
3. Use the random module to randomly choose the ship's row and column.
4. Do not use global variables. Pass values as arguments to your functions instead.
5. Do not create any extra functions or add any extra parameters.
6. Use docstrings (triple-quote style) to document each of your functions. Describe all parameters and return values.
7. Add brief comments (# style) within your functions to describe sections of code.
8. Thoroughly test your program before submitting/demoing.
    a. Make sure your main code is in a main function.
    b. Make sure each of your functions passes in the correct parameters and returns the correct values.
    c. Make sure to validate all user input.
    d. Make sure that your grid is 5x5, not 6x6.
    e. Make sure the board is updated correctly after the user makes their guess.
    f. Make sure that the user cannot win by choosing the same location 4 times.
    g. Make sure that the board is reset and solution is re-randomized whenever the user wins or they have chosen to show the solution.

Battleship – Time estimate: 4 hours

Battleship **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Reset Function: 1. Resets game board. 2. Randomizes location of ship. 3. Returns the board and solution.					
Display Func and Get Row Func: 1. Passes in 5x5 game board. 2. Displays the contents of the board. 3. Displays with column header 1-5. 4. Displays with row header A-E. 5. Prompts user for row A-E. 6. Returns valid input.					
Fire Shot Function: 1. Passes in game board, solution, and user's row and col. 2. Places 'x' on game board if user's guess is a miss. 3. Places '*' on game board if user's guess is a hit. 4. Returns true if user's guess was a hit, otherwise false.					
Main & Output: 1. Initializes game board and solution using reset_game. 2. Repeatedly prompts user with main menu. 3. Prompts user for row/col input. 4. Checks that user's guess wasn't already inputted. 5. Correctly displays updated grid. 6. Correctly displays when user wins. 7. Correctly displays solution (opt 2). 8. Resets game after win/opt 2.					
Code Formatting: 1. Main code is in a main function. 2. Correct spacing. 3. Meaningful variable names. 4. No global variables. 5. Correctly documented.					