CECS 277 – Lab 4 – File IO

Treasure Hunter

Create a program that allows the user to wander through a map in search of treasure while trying to avoid traps. The user starts at the upper left corner of the map and then they can move up, down, left, or right, as long as they stay within the boundaries of the 7x7 map. The user can receive a hint that will tell them the number of treasures and traps that are nearby. If the user finds all seven treasures, then they win, if they run into any traps, then they lose.

Create the following functions for your program:

1. `read_map()` – read in the contents of the file ('map.txt') and store the contents in a 2D list. Each character in the maze (after removing the spaces and new lines) should be stored as a separate element in the 2D list. Return the filled 2D list.
2. `display_map(map, player)` – passes in the map and the user's location. Iterate through the contents of the maze. Display each character in the maze in a matrix format. When you reach the user's location, display a 'P' instead so that it shows where the user currently is in the maze (do not actually place the 'P' in the 2D list, just display it).
3. `move_player(player, dir, upper_bound)` – moves the player in the selected direction (W=up, A=left, S=down, D=right). The boundaries of the map are between 0-upper_bound. Check that the location that the user is trying to move to isn't out of bounds. If it isn't, then update the user's location by changing the row and column values in the location list by adding or subtracting 1 to the row or column (depending on the direction they moved), otherwise, display an error message and do not update the user's location.
4. `count_treasures_traps(map, player, upper_bound)` – iterates through the surrounding spaces of the user's current location. Keep a count of the number of treasures ('T'), and traps ('X') that are in those spaces. Return the two counts.

In the main function, call the read_file function and store the result in a 2D list. This is the solution map that is hidden from the user and is used to determine if they have found a treasure or a trap. Create a second 2D list filled with dots ('.'), this will be the map that is displayed to the user and will be updated to show the hints and any treasures that they have found so far. Create a two item 1D list for the user that stores the row and column of their location in the maze and is initialized to the upper left hand corner of the map (0, 0). Use a loop to display the user's map and then prompt the user to move in a direction, to ask for a hint, or to quit the game. Move the user by calling the move_player function and then check to see if they found a treasure or a trap. If they have found a treasure, then mark a 'T' on the user's map and display the number of treasures remaining. If they have found a trap, then the user loses, and the game ends. If they find all seven treasures, then the user wins and the game ends. If the user chooses 'L' to look around (get a hint), then call count_treasure_traps to get the number of treasures and traps in the spaces surrounding the player's current location. Display the number of treasures and traps to the user, and then update the user's map to display the number of traps that surround that space (unless there is already a 'T' there). If the user chooses 'Q' to quit, then the game ends. and do not update the user's location.

Example Output:

```
Treasure Hunt!
Find the 7 treasures without getting
caught in a trap.  Look around to spot
nearby traps and treasures.
P . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  x
Invalid input.
P . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  w
You cannot move that direction.
P . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  d
You found treasure!
There are 6 treasures remaining.
. P . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  L
You detect 1 treasures nearby.
You detect 0 traps nearby.
. P . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  s
. T . . . . .
. P . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  L
You detect 1 treasures nearby.
You detect 1 traps nearby.
. T . . . . .
. P . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  d
. T . . . . .
. 1 P . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  L
You detect 2 treasures nearby.
You detect 1 traps nearby.
. T . . . . .
. 1 P . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .

Enter Direction (WASD or L to Look around
or Q to quit):  d
. T . . . . .
. 1 1 P . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  L
You detect 1 treasures nearby.
You detect 1 traps nearby.
. T . . . . .
. 1 1 P . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Enter Direction (WASD or L to Look around
or Q to quit):  w
You were caught in a trap!
You found 1 treasures.
. T . P . . .
. 1 1 1 . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
Game Over!
```

Notes:

1. Please place your name, date, and a brief description in a comment block at the top of your program.
2. Do not create any extra functions or add any extra parameters.
3. Please do not create any global variables, instead, pass variables as arguments to the functions and return values back when needed.
4. Please read through the Coding Standards reference document on Canvas for guidelines on how to name your variables and to format your program.
5. Use docstrings to document each of your functions. Document all parameters and return values. Add brief comments in your program to describe sections of code.
6. You may make your own maps if you like, but they should be 7x7 and have 7 treasures.
7. Thoroughly test your program before submitting:
    a. Make sure that the file is read in correctly and each character is stored in a separate element in the 2D list (no spaces or new lines in the 2D list).
    b. Make sure that you don't mix up the rows and columns of the 2D list.
    c. Make sure that the user starts at the top left corner (0,0).
    d. Make sure that your map is displayed correctly (ie. properly oriented).
    e. Make sure that the user's location is displayed in the correct location on the map, but the 'P' is not added to the 2D list
    f. Make sure that all user input is checked for invalid values.
    g. Make sure that the player correctly moves in the direction the user specified.
    h. Make sure that the player cannot move through walls in any direction.
    i. Make sure that the game correctly detects when a treasure or trap is discovered.
    j. Make sure that the map is correctly updated (ex. a 'T' is placed at locations where a treasure was found, and numbers are placed where player asked for a hint and is the number of traps that are in the surrounding 8 spaces of the player).
    k. Make sure that if the player asks for a hint when standing on a treasure, the 'T' should stay on the map (ie. 'T' has precedence over the number of hints), but the program should still output the number of treasures and traps nearby.
    l. Make sure that the program ends when the user finds all seven treasures, if they get caught in a trap, or when the user quits the game.

Treasure Hunter Rubric – Time estimate: 4 hours

Treasure Hunter **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
read_map function: 1. Creates a 2D list. 2. Opens file and reads in contents of the file. 3. Stores file contents into 2D list where each character in the map is a separate element. 4. Returns filled 2D list.					
move_player and display_map functions: 1. move_player adds/subtracts 1 to user's row/col to update the players's location on map. 2. Does not move user out of bounds of map. 3. Displays map as a grid. 4. 'P' is placed at user's loc. 5. Does not add 'P' to the list.					
count_treas_traps function: 1. Iterates through spaces surrounding the players loc. 2. Counts treasures and traps in those spaces. 3. Returns the pair of counts.					
Main Function: 1. Creates two maps. 2. Initializes user's loc at (0,0). 3. Displays map and menu. 4. Checks user's input. 5. Updates user's location based on user's choice. 6. Cannot move out of bounds. 7. 'L' detects treasures/traps. 8. User map is updated. 9. Repeats until win/loss/quit.					
Code Formatting: 1. Code is in functions. 2. Correct spacing. 3. Meaningful variable names. 4. No global variables. 5. Correctly documented.					