Introduction to Programming

CMPT 120

Project 3 – Extending the Design – 100 points

Deadline

Final push no later than 8AM Monday on 3 November 2023

Goals

To continue development of your "robot" simulator term project by adding obstacles and goals to the environment, along with error handling and saving the program state to a file. To practice top-down design and the use of functions as an essential part of the design process.

Preparation

For this assignment, you must have already completed the second version of your project hosted in the *{YourLastname}-Work* repository of our class organization on GitHub.com. Open your Python script in the VSCode IDE to continue editing your code.

Instructions

Create the next version of your project by implementing the required features listed below.

★ *Requirement #1 – Use emoji characters for the walls, empty spaces, and robot.* **10pts**
- Instead of "#", " ", and "R", use emoji characters such as 🧱, ⬛, and 🔼. For the robot, be sure to use appropriate characters to represent each direction.
- Tip: Declare global variables to hold the different string values that you will use to represent elements of the room such as the walls, the robot, empty space, etc.

★ *Requirement #2 – Use a 2D matrix to represent the room and its contents.* **20pts**
- Instead of simply storing the room width and height, create a list-of-lists (that is, a 2D list, or matrix) to hold data about what is inside the room at each x-y location. Initially, the room will simply be empty.
- After initializing the robot's coordinates, place the robot in the room matrix. When moving the robot, you will place the robot in its new location but be sure also to set the robot's previous location back to a blank cell.

★ *Requirement #3 – Add obstacles to the room.* **20pts**
- After you select the robot's starting location, place several obstacles into the room matrix at random locations. The number of obstacles should depend on the size of the room, with an average of one obstacle for every 10 spaces/cells in the room.
- You must ensure that no obstacle is placed on top of the robot.

★ *Requirement #4 – Add a goal location to your room.* **15pts**
- Before choosing the robot's starting location, randomly choose coordinates for the mission goal – the location that the robot must reach to complete its mission – and place this goal in the room matrix. When choosing the robot's start location, make sure that you do not place it on top of the goal at the start.

★ *Requirement #5 – Prevent the robot from moving onto obstacles.* **15pts**
- Before moving the robot, check that the target location does not contain an obstacle. Depending on how you are already checking for walls, you might be able to use the same approach for obstacles.

★ *Requirement #6 – End the simulation when the robot reaches the goal.* **20pts**
- Whenever the robot moves, check to see if it has moved onto the goal location.
- If it does, then display a congratulatory message and break out of the loop.

Advice

Follow coding best practices with mnemonic naming, consistent indentation and spacing, frequent comments., and appropriate use of user-defined functions.

Be sure to stage your changes a little at a time and commit frequently. Push any local work to your remote GitHub repository regularly. Don't forget to:

- Write short but meaningful messages for every commit. (Tip: Consider using the instructions themselves as your commit messages.)
- Look at the differences between successive versions of your code. You can do this on the GitHub website as well as in GitHub Desktop app itself.

Test, test, test… and test again. Then test some more. When you think you've tested enough, go back and test yet again. Then get someone else to test for you while you test theirs. Etc.

Submitting

You must push your changes to GitHub before the due date.

Note: Pushing regularly will reduce the risk of losing your work, so do not wait until after you have made all changes and commits before pushing.