CMPT 310. Winter 2024, Assignment # 1:

Introduction:
The goal in this assignment is to practice design and implementation of the common search algorithms for intelligent agents in a static fully observable grid environment. This assignment is provided in the form of a shell module, and you are going to add or modify it to fulfill the requirements. The parts that you will need to implement have printout comments saying "Your code goes here" or such. Once you have done a part, comment out the print function using #.

The code base distributed is a shell with all the mechanics been implemented. The code contains lots of comments and instructions. Read them carefully. They are part of the assignment description.

The environment is a 2D grid of rooms among which some are set to 'dirty'(grey color). There is a collection of blocked rooms as well. The blocked rooms and the boundary walls are red colored. There is a vacuum cleaning agent which moves from room to room and cleans them. The agent has a location and direction that is heading. The default cost of moving from room to room is 1.

Tasks:
The goal is to use various search algorithms to help the robot to go around efficiently and clean rooms, as intelligently as possible.
You are going to implement 5 search algorithms: BF Graph, DF Graph, UCS, Greedy, and A*. The tasks are basically to complete and implement all the algorithms using the framework provided.

The way search will work is that the path node and the explore list are computed when one selects a search algorithm from the menu. The explored list of nodes is immediately displayed using pink color. Using the 'next' or 'run' buttons one can step through the path from the current agent location to the selected goal which is a dirty room. For 'run' the steps follow one another automatically. The app displays 2 counters at the top, keeps track of the total number of *explored* rooms and the number of rooms on *path* as the agent progresses through the grid automatically.
Use the RomaniaMap Example as your guidance through implementing the searches

Commandline Arguments:
The agent can be launched from command line as following:

```
>python VacuumSearch.py -s searchType -c costFunction -r heuristic -n (corners)
```

The detail description of each option is described in function readCommand() in script vacuum_search.py. searchType and costFunction options can be set using the GUI, but the last 2 options in this command can only be set using commandline launch or using the run configuration provided with the project.

HandIN: Put all the edited py scripts and your analysis file (part 6 bellow) in a folder and zipped it. Name the file as **LastName_studentNum_ass1.zip** and upload to Canvas.

1- **Basic Implementation:** Implement basic search algorithms BFS, DFS, UCS, Greedy, and A* with default setting meaning the cost function being number of steps, and Heuristic being Manhatttan distance. Each algorithm is worth **10** points, for total of **50** poins.
You can test each algorithm from GUI. For the correctness test use the corresponding Run configuration from the top menu in PyCharm to launch the app. Then select Run. You should get the foll numbers for the counters:
   - BFS: ExploredCount=478, PathCount=59
   - DFS: ExploredCount=647, PathCount=350
   - UCS: ExploredCount=545, PathCount=59
   - Greedy: ExploredCount=113, PathCount=78
   - A*: ExploredCount=154, PathCount=59

   Notice how BFS and A* and UCS have the same PathCount.

2- Above you have already implemented findMinManhattanDist() which is used for Greedy and A*. This part is worth **10** points. To test the effect of heuristic implement findMinEuclidDist() and see the difference. Only informed searched would be affected by this change. FindMinEuclidDist() uses Squared of Euclid distance function which is implemented in Utils.py. The change of counts would be minimal: The difference of counts that you should observe is reported in the attached list at the end of this file.

3- Path cost function affects all the algorithms above except Greedy. The default cost is just the number of steps from the starting point. We want to make our agent pay for any turn it makes. You are going to implement computeTurnCost() which adds 50% of a step cost for each 90′ turn to total cost for each action. This means for example if the agent is currently heading north and the current action is LEFT, it is going to pay 0.5 of a move cost. If the action is DOWN, then the turn cost will be 1, and if the action is RIGHT, the turn cost will be 0.5. The part is worth **20** points.

4- We want to encourage our agent to stay on the left of the grid more. This can be done by devising a cost function that promotes such a behavior. Implement this cost function and link it to the current cost option in the menu ‘StayLeft’. You can test it by running the option from GUI. This is worth **10** points.

5- Repeat the above step for ‘StayTop’ cost function as well. This should encourage the agent to clean the dirty rooms in upper part of the grid first.

6- Analysis (**10** points). As the last part of this assignment, test your solutions for informed searches and various cost function for a grid where there are only 4 dirty rooms in 4 corners. Run configurations below to verify the count numbers are correct:

a.	Greedy_Corners	79	72
b.	A*_Corners	104	72

   In the second test,b, if you change cost function from Step to StepTurn, what do you expect to get for the counts? What did you get? Interpret and explain the differences between a, b, and the switch of cost function.

```
Test verification results:
ucs stayUp             679  95
Algorithm   Heuristic  CostFunction     ExploredCount     PathCount
A*          Manhattan  StepCount        154               59
A*          Euclid     StepCount        101               72
Greedy      Manhattan  StepCount        113               78
Greedy      Euclid     StepCount        100               76
BFS                    StepCount        478               59
DFS                    StepCount        647               350
Greedy Corners         StepCount        79                72
A* Corners Manhatten   StepCount        104               72
A* Corners Euclid      StepCount        78                72
UCS                    StepTurn         662               96
UCS                    StepCount        545               59
```