CMPE540 – 2022 Spring Homework 1 – Plan a path for the vacuum cleaner
Due: April 7th, midnight (strict)

You will implement single-agent search in this homework. Submit a single .py file and a single .pdf file.

The program will be called with python3.9 in Linux (Ubuntu) with:

```
python3 <lastname.py> <search-type> <init-file>
```

where <search-type> might be one of the following:
- DFS
- BFS
- UCS
- GS
- A*1
- A*2

<init-file> will be a text file gives all details related to the initial environment.

Submission Instructions:
- Please follow these instructions, otherwise you will lose significant points.
- The project will be automatically evaluated. Still, you need to prepare a very short report. See the end of this document for more information about what is expected in the report.
- Name your file with your lastname (lastname.pdf lastname.py)

Late Submission:
- - (10 x days) upto 3 days
- In case there is serious mistake in the project description or significant change, this date will be postponed.

Discussion and Cheating:
- I don't like this section, but still should write about this..
- You are welcome to discuss the topics about the project. But please do not cheat. There are programs that can automatically check the similarity of the source code and/or execution. Read the department policy about what is considered as cheating. If I am convinced that you cheated, then you will get F and disciplinary action is filed.

Project objectives:
- In this project, you will
  - implement basic uninformed search algorithms:
    - breath first search (BFS),
    - depth first search (DFS),
    - uniform cost search (UCS)
  - implement basic informed search algorihm:
    - A* search
  - design states based on given problem descriptions
  - design admissible and consistent heuristics that will be used by A* search

Introduction to the problem:

The environment is as follows:
- The environment is NxM grid world.
- Each grid in the environment might contain:
  - Vacuum cleaner (our agent).
  - Obstacles that avoid entering to that grid. There is not dirt in the obstacle with grid.
  - One or more dirts (max 9 dirts)
  - "Jumper" which moves the agent that moves an incoming agent to the next grid (if the next grid does not contain an obstacle). For example, an agent coming from left to a jumper grid is transformed to the grid on the right. An agent coming from up is transformed to the grid down.
- The vacuum cleaner has five actions:
  - left, right, up, down moves the cleaner one grid, unless that grid is an obstacle.
  - suck action that sucks one dirt. (in order to clean n dirts in a grid, suck action should be executed n times)
- Costs of the actions:
  - Left and right: 1
  - Up and down: 2
  - Suck: 5

- The aim is to apply
  - DFS, BFS, UCS, GS, A* search to clean the grid world.
- The environment, agent type, locations of the obstacles, dirt, vacuum cleaner (our agent) and jumper will be provided in a text file.

- For A* search:
  - For A*1, use the Manhattan Distance to the closest dirt as the heuristics function.
  - For A*2, design your own heuristic

- Tie-breaker:
  - Tie situations might occur during inserting into the fringe. The precedence used for fringe insertion tie-breaker is as follows: suck, left, right, down, up
  - For DFS and BFS, how to remove the nodes from the fringe is well-defined.
  - For others, if costs/g-values/f-values are same, respect the insertion order (first-in first-out).

Example input file:

```
xxxxxxxxxx
x  1   1 x
x   x  j x
x j x  2 x
x c x    x
xxxxxxxxxx
```

- where
  - x corresponds to obstacles
  - c corresponds to the vacuum-cleaner
  - <digit> corresponds to the number of dirts in the corresponding grid
  - j corresponds to the jumper

Output:

- You need to report (to standard output):
  - the number of expanded nodes (in the first line)
  - the action sequence to achieve the goal (in the second line)
  - the heuristic function value if the <search-type> is A*2. (in the third line)

Report:

- Describe how state is defined with 2 sentences most.
- How many states are there? (Show calculation)
- Describe the heuristic function designed for A*2. Make sure your description is clear and easy to understand.