Project 1
Artificial Intelligence
CSCE 5210 – Fall 2021
Distributed: Tuesday, August 31
Due: Tuesday September 21

[Solutions to this assignment must be submitted via CANVAS prior to midnight on the due date. Submissions no more than one day late will not be penalized. Submissions up to one week late will be penalized 10 points. Submissions will not be accepted if more than one week later than the due date.]

This project is to be undertaken in pairs. No individual work please. Please state clearly the names of the two people undertaking the project and the contributions that each has made. Only one submission should be made per group.

Purpose: To gain a thorough understanding of the working of a simple, table-driven, reflex agent. This will enable you to put into practice the concepts of performance measurement, environment interaction, and the use of actuators (PEAS) to assist the agent's navigation in a stochastic setting that mimics a real-world environment.

Environment Description: The environment is a warehouse represented by a 6 by 6 grid. Each grid can either be empty (i.e. a passage that leads to a shelf) or is a shelf. There are 10 shelves in the warehouse, labelled A, B, …, J. Each shelf contains a collection of items of a particular type. Thus, for example, shelf A contains multiple items of type A. The objective is to service customer orders through the use of a robot equipped with the agent software and hardware sensors that are mounted on the robot.

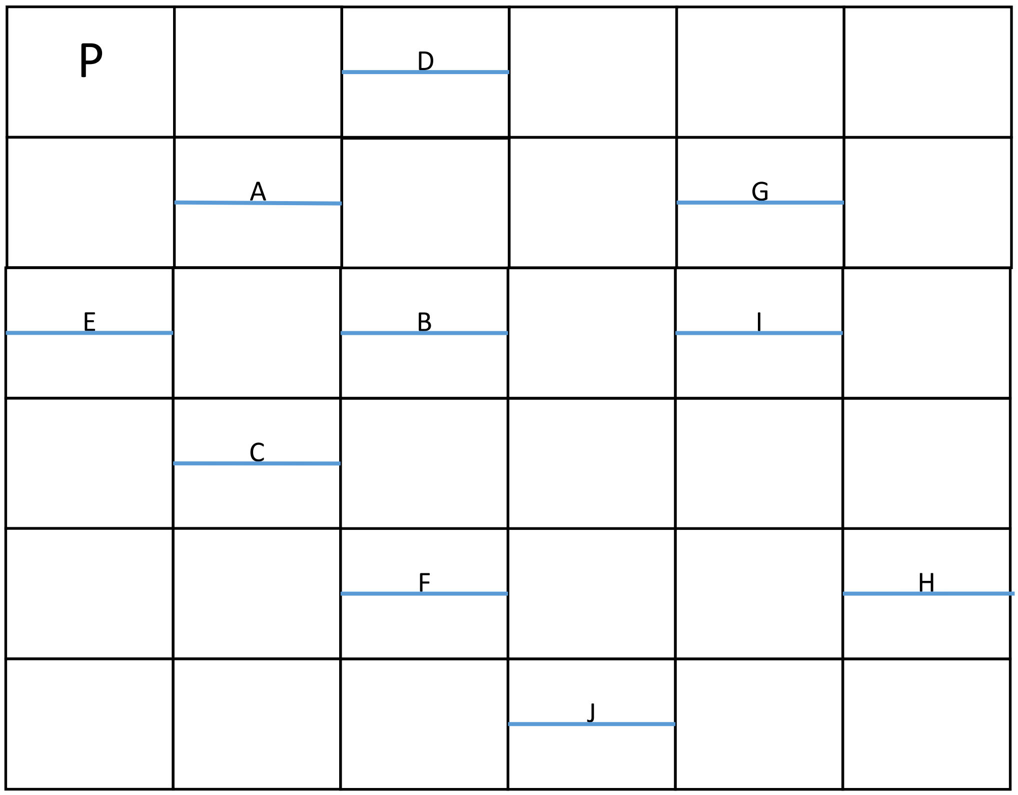
The sales ordering system accepts customer orders which are relayed to the robot wirelessly. The number of items in a given order can be taken as a random number between 1 and the total number of item types (i.e. shelves). When a robot receives a new order, it retrieves the items ordered by visiting the shelves that hold the items ordered. In order to perceive the location of shelves the robot uses sensors that are 80% accurate. Thus 80% of the time the robot will correctly sense whether a shelf exists in its neighborhood. Assume that the neighborhood consists of a maximum of 4 grid positions which are to the left, right, up and down from its current position. When the robot senses a shelf, it can move to the shelf and extract the type of item that is stored in that shelf. However, 10% of the time the sensor fails, and the robot thinks that a shelf is present when it is not the case. Similarly, 10% of the time a shelf exists but the sensor fails to detect it until it lands on the grid containing the shelf.

The agent's memory is limited to the current episode, which is the path plotted to retrieve the items for the order being serviced. It has no memory of previous episodes.

The agent should use the following rules to navigate:

1. If any of the grid positions in its neighborhood refer to an item ordered, then it moves to that grid position. If two or more neighboring positions contain ordered items, then the tie is broken by making a random choice between the positions involved.
2. If none of the neighboring positions contain ordered items, then a random choice is made to a position that has not been visited before.

Each visit to a position in a grid that does not contain an ordered item incurs a score of -1, while each visit to a position containing an ordered item scores +3. Thus a brute force algorithm that sweeps through the grid from the starting position P will score a total score of (35-n)*(-1)+n*3=-35+4*n where n is the number of items ordered. Since the robot can sense neighboring positions, your agent should do better than a brute force method when its score is averaged over a large number (say 1000) episodes.



Run your agent for 1000 episodes and print its average score. You may assume that the stock is adequate for 1000 episodes and so stock never runs out. Once an episode is over the robot immediately starts a new episode for a new order if one exists, otherwise it waits in its current position for the next order to arrive.

You are required to write a Python program to implement the agent controlling the robot.

Your program should produce the following:

1. The average score taken across 1000 episodes.
2. Display the shortest path (i.e., the sequence of grid positions visited) across the 1000 episodes and its corresponding score.
3. Display the longest path across the 1000 episodes and its corresponding score.
4. Demonstrate that your program is generic enough to cope with a new layout of shelving as shown below. This should be done by carrying out requirements 1, 2 and 3 above for the new layout.
5. Give two possible reasons why the average score differs from the average score computed across the old layout.
6. What data structure(s) did you use to ensure that your program can cope with any given layout?

P		A		P	
D		B		M	
E		F		K	
C		H			O
G		J			Q
I				N	

Hand in:

1) A short report (between 1 and 2 pages in size) describing the approach that you took. In addition to the description of your approach you should answer the 6 questions given above.
2) Any assumptions that you made while undertaking this project.
3) The complete set of Python code that you used.