Chapter 2, Ex 2.3

1. An agent that senses only partial information about the state cannot be perfectly rational.
    a. False. Rationality means based on the information or the knowledge provided, the agent gives the maximum measure of performance. For example, an automated vehicle requires environmental information from only the specific sensors based on the functionality. If a vehicle has automated guided direction then even without information like temperature and wind speed of the state it's able to achieve the task rationally.
2. There exist task environments in which no pure reflex agent can behave rationally.
    a. True. Because a pure reflex agent will not base its results on the previous output but on the current state of the environment. So if a machine requires an output which is based on the previous output and the current state then it is not possible by a pure reflex agent. So, in this case a pure reflex agent cannot behave rationally.
3. There exists a task environment in which every agent is rational.
    a. True. Consider an environment where all the agents are made to give out only one output for each and every environment state. It's like a very simple, not so smart, basic machine.
4. The input to an agent program is the same as the input to the agent function.
    a. False. This is not true as the agent function takes all the percept input that has happened whereas the agent program takes in only the current percept.
5. Every agent function is implementable by some program/machine combination.
    a. False. The above statement is not true in cases where the problem doesn't have a solution or it's not feasible to construct the problem into some program.
6. Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.
    a. True. Similar to question (c), this could be possible when the agents have the same output. Or also another possibility could be that the main task is to select randomly. For example, selecting a toy randomly by a toddler.
7. It is possible for a given agent to be perfectly rational in two distinct task environments.
    a. True. Let's take an example. There are two environments, one records a person's intelligence level and the other records his swimming skills. If there is a function looking for a math tutor then it would give priority to the person with a higher intelligence whereas another function that is looking for a swimmer will give priority to a person with a higher swimming skill. So in both the cases the agent will act rationally.
8. Every agent is rational in an unobservable environment.

a. False. For example if a grass trimmer agent is made to just move without actually cutting then it's not rational. So it’s not true with every agent.

9. A perfectly rational poker-playing agent never loses.
    a. False. If there are two rational agents in a poker game then one of them has to lose.

Chapter 3, Ex 3.2

Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

A. Formulate this problem. How large is the state space?
    a. State space is the initial state, the possible actions that can be taken to reach the final goal and the transition model. It can be represented in the form of a graph where the nodes are the states and the connecting path is the action or the cost of the action.
    Initial State: centre of the maze facing north.
    Goal State: Anypoint just outside the maze walls.
    Actions: Face north, south, east or west and move.
    State space: There are 4 possible directions to move and let’s take n as the number of locations on the maze. So in the simplest way we can calculate the state space as 4*n.

B. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?
    a. Let's take the number of intersections in the maze as i. Total number of blocks remains the same, ie n.
    b. Now this can be divided into parts. One, possible actions the robot can do when at the intersection (i) and second, possible actions it can do when not at the intersection, which is the remaining (n-i) blocks.
    c. At the intersection it can move to north, south, east or west. So in that case the it will be 4*i. Whereas on other blocks it can either move forward or stop on reaching a wall. So it will be 2(n-i).
    d. Hence, the final state space can be calculated as 4i + 2(n-i).

C. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot’s orientation now?

a. In this case an action is taken only when the robot hits a wall or comes at an intersection. The only possible action that needs to be taken is to turn. So not problem can be formulated as 1*i, where i is the number of intersection blocks in the maze.
b. And as all the moving is done by the robot automatically, there is not need to keep track of the orientation of the robot.

D. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.
a. There was very little information on the design of the maze. Like what if the robot gets stuck in a deadend. Or if there are any obstacles other than the wall.
b. We don't have a clarity on how the robot functions in the sense of direction. It's assumed that it can only go north, south, east or west. What about moving diagonally?
c. Consideration of the environment was not taken into account. Variables like the temperature, wind and other natural causes that could affect the outcome of the robot were ignored.

Chapter 2, Ex 2.9

```
def vacuum_world():

    goalState = {'A': 0, 'B': 0}
    cost = 0

    inputLoc = input("Enter the initial Location of the Vacuum (A or
B): ")
    inputEnvState = input("Enter environment state of " + inputLoc + "
(1 for dirty and 0 for clean): ")
    otherEnvState = input("Enter environment state of other room (1 for
dirty and 0 for clean): ")

    if inputLoc == 'A':
        print("Vacuum is initially at Location A")
        goalState = {'A': inputEnvState, 'B': otherEnvState}
        print("So the starting state is as follows: " + str(goalState))
        if inputEnvState == '1':
            print("Location A is Dirty.")
            goalState['A'] = '0'
            cost += 1
            print("Cost for cleaning A: " + str(cost))
```

```
            print("Location A has been Cleaned.")

            print("Moving right to the Location B. ")
            cost += 1
            print("Cost for moving right: " + str(cost))

            if otherEnvState == '1':
                print("Location B is Dirty.")
                goalState['B'] = '0'
                cost += 1
                print("Cost for cleaning B: " + str(cost))
                print("Location B has been Cleaned. ")
            else:
                print("No action required")
                print("Location B is already clean.")

        if inputEnvState == '0':
            print("Location A is already clean ")
            print("Moving right to the Location B. ")
            cost += 1
            print("Cost for moving right: " + str(cost))
            if otherEnvState == '1':
                print("Location B is Dirty.")

                goalState['B'] = '0'
                cost += 1
                print("Cost for cleaning B: " + str(cost))
                print("Location B has been Cleaned. ")
            else:
                print("No action required")
                print("Location B is already clean.")

    else:
        print("Vacuum is initially at Location B")
        goalState = {'B': inputEnvState, 'A': otherEnvState}
        print("So the starting state is as follows: ")
        print(goalState)
        if inputEnvState == '1':
            print("Location B is Dirty.")
            goalState['B'] = '0'
            cost += 1
            print("Cost for cleaning B:  " + str(cost))
            print("Location B has been Cleaned.")
```

```
            print("Moving left to Location A. ")
            cost += 1
            print("Cost for moving left: " + str(cost))
            if otherEnvState == '1':
                print("Location A is Dirty.")
                goalState['A'] = '0'
                cost += 1
                print("Cost for cleaning A: " + str(cost))
                print("Location A has been Cleaned.")
            else:
                print("No action required")
                print("Location A is already clean.")

        else:
            print("Location B is already clean.")
            print("Moving left to Location A. ")
            cost += 1
            print("Cost for moving left: " + str(cost))
            if otherEnvState == '1':
                print("Location A is Dirty.")
                goalState['A'] = '0'
                cost += 1
                print("Cost for cleaning A: " + str(cost))
                print("Location A has been Cleaned. ")
            else:
                print("No action required")
                print("Location A is already clean.")

    print("Reached the goal state: "+ str(goalState))
    print("Final performance Cost: " + str(cost))


vacuum_world()
```

Example input and its respected output:

Input1:

{‘B’: 1 , ‘A’: 1}

Output1:

```
Enter the initial Location of the Vacuum (A or B): B
Enter environment state of B (1 for dirty and 0 for clean): 1
Enter environment state of other room (1 for dirty and 0 for clean): 1
Vacuum is initially at Location B
So the starting state is as follows:
{'B': '1', 'A': '1'}
Location B is Dirty.
Cost for cleaning B:  1
Location B has been Cleaned.
Moving left to Location A.
Cost for moving left: 2
Location A is Dirty.
Cost for cleaning A: 3
Location A has been Cleaned.
Reached the goal state: {'B': '0', 'A': '0'}
Final performance Cost: 3
```

Goal State = {‘A’: 0 , ‘B’: 0}
Cost is incremented every time the vacuum cleaner sucks and everytime it moves from A to B or vice versa.

Initial State	Total cost to reach goal state
{‘A’: 0 , ‘B’: 0}	1
{‘A’: 0 , ‘B’: 1}	2
{‘A’: 1 , ‘B’: 0}	2
{‘A’: 1 , ‘B’: 1}	3
{‘B’: 0 , ‘A’: 0}	1
{‘B’: 0 , ‘A’: 1}	2
{‘B’: 1 , ‘A’: 0}	2
{‘B’: 1 , ‘A’: 1}	3

Average of the cost = (1+1+2+2+2+2+3+3)/8 = 2

Chapter 3, Ex 3.3

A. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

   Let i and j be the cities that the two friends are currently in. Let friend 1 be p1 and friend 2 be p2. i and j can be the same city if p1 and p2 are in the same place.
   a. Initial State: i, j where p1 is in city i and p2 is in city j
   b. Goal State: (i,j) where i and j is the same city.
   c. Action: Action that can be taken is going from state (i,j) to (x,y) where x and y are the adjacent cities to i and j.
   d. State space: All the possible pairs of (i,j).
   e. Cost: moving from state (i,j) to (x,y) its max(d(i, x), d(j, y)). Where d(i, x) is the distance between the cities i and x.

B. LetD(i, j)be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible? (i)D(i, j); (ii)2·D(i, j); (iii)D(i, j)/2.
   a. A heuristic function is said to be admissible if it has the most optimal solution. In this case the most optimal solution would be if p1 and p2 walked towards each other and met halfway to each other. Hence (iii) D(i,j)/2 is an admissible heuristic function.

C. Are there completely connected maps for which no solution exists?
   a. Yes it is possible to have a completely connected map with no solutions. For example if p1 and p2 are in i and j cities which are adjacent to each other.
      P1,i ---------- p2,j to p2,i ---------- p1,j (completely connected graph)

D. Are there maps in which all solutions require one friend to visit the same city twice?
   a. Yes, it's possible.
   b. Below shows a similar example of the trajectory of p1 and p2 that meet at city C. In this case p2 crosses the same city twice to reach the goal state.