COEN 166 Artificial Intelligence

Lab Assignment #2: Vacuum Cleaner Agent Report

Gagan Gupta **SCU#00001478479**

Explanation of the Functions:

__init__:

Constructor for the Vacuum Class that takes in the state array as input and initializes the state array to that value, the cost counter to 0, and the actions array to an empty array.

Suck:

Has no input and output. If the vacuum is on the left side, it will clean the left side. Else, it will clean the right side. It then appends the action to the action list and increases the cost by one.

Right:

Has no input and output. Moves the robot to the right side if on the left side. It then appends the action to the action list and increases the cost by one.

Left:

Has no input and output. Moves the robot to the left side if on the right side. It then appends the action to the action list and increases the cost by one.

RunActions:

Has no input and output. If left is dirty and robot is on left, then suck and call runActions again. If right is dirty and robot is on right, then suck and call runActions again. If left is dirty and robot is on right, then go left and call runActions again. If right is dirty and robot is on left, then go left and call runActions again. In the end if neither of the squares are dirty then the function is done.

Explanation of the test cases:

TEST CASE where state=[‘C’,’C’,’L’]

Status of Left side:

C

Status of Right side:

C

Location of Vacuum:

L

List of Actions: []

Ending Location of Vacuum: L

Cost of the State inputted: 0

EXPLANATION: No actions were taken as both squares were already clean to begin with. In total this is 0 actions therefore our cost is 0.

TEST CASE where state=[‘D’,’D’,’R’]

Status of Left side:

D

Status of Right side:

D

Location of Vacuum:

R

List of Actions: ['Suck on Right', 'Move to Left', 'Suck on Left']

Ending Location of Vacuum: L

Cost of the State inputted: 3

EXPLANATION: The current square (right) is dirty so we first clean that. We see that left is dirty, so we move to the left. The current square (left) is dirty so we clean that. In total this is 3 actions therefore our cost is 3.

Appendix:

```
class Vacuum:
    def __init__(self, state):
        self.state=state
        self.cost=0
        self.actions=[]
    def suck(self):
        if self.state[2]=='L':
            self.actions.append("Suck on Left")
            self.state[0]='C'
        else:
            self.actions.append("Suck on Right")
            self.state[1]='C'
        self.cost = self.cost + 1

    def right(self):
        if self.state[2]=='L':
            self.actions.append("Move to Right")
            self.state[2]='R'
        self.cost = self.cost + 1

    def left(self):
        if self.state[2]=='R':
            self.actions.append("Move to Left")
            self.state[2]='L'
        self.cost = self.cost + 1

    def runActions(self):
        if self.state[0]=='D' and self.state[2]=='L':
            self.suck()
            self.runActions()
        if self.state[1]=='D' and self.state[2]=='R':
            self.suck()
            self.runActions()
        if self.state[1]=='D' and self.state[2]=='L':
            self.right()
            self.runActions()
        if self.state[0]=='D' and self.state[2]=='R':
            self.left()
            self.runActions()
```

```
def main():
        state=['','','']
        while state[0]!='C' and state[0]!='D':
                print("Status of Left side: ")
                state[0] = str(input())
        while state[1] != 'C' and state[1] != 'D':
                print("Status of Right side: ")
                state[1] = str(input())
        while state[2]!='L' and state[2]!='R':
                print("Location of Vacuum: ")
                state[2] = str(input())
        x=Vacuum(state)
        x.runActions()
        print("List of Actions: ",x.actions)
        print("Ending Location of Vacuum: ",x.state[2])
        print("Cost of the State inputted: ",x.cost)

if __name__ == "__main__":
   main()
```