COEN 166 Artificial Intelligence

Lab Assignment #2: Vacuum Cleaner Agent - Sample Report

Name: Eva Stenberg   **ID: 00001576722**

Explanation of the defined functions:

1. Function def agent_vacuum : This function is the goal state which runs a while loop through all 8 possible states, counts the actions taken for each state, and returns the action count

2. Function def run: The function takes in the state of the location and determines the actions. If the space is dirty then it sucks, else it moves either left or right depending if the current space it is in is dirty.

Explanation of the test case:
The test case is created to test whether the program follows the correct actions of the vacuum cleaner agent.

Starting state: ['Clean', 'Clean', 0]
Number of actions: 0
Actions taken: []
Correct Outputs!

Starting state: ['Clean', 'Clean', 1]
Number of actions: 0
Actions taken: []
Correct Outputs!

Starting state: ['Clean', 'Dirty', 0]
Number of actions: 2
Actions taken: ['Right', 'Suck']
Correct Outputs!

Starting state: ['Clean', 'Dirty', 1]
Number of actions: 1
Actions taken: ['Suck']
Correct Outputs!

Starting state: ['Dirty', 'Clean', 0]
Number of actions: 1
Actions taken: ['Suck']
Correct Outputs!

Starting state: ['Dirty', 'Clean', 1]
Number of actions: 2
Actions taken: ['Left', 'Suck']
Correct Outputs!

Starting state: ['Dirty', 'Dirty', 0]
Number of actions: 3
Actions taken: ['Suck', 'Right', 'Suck']
Correct Outputs!

Starting state: ['Dirty', 'Dirty', 1]
Number of actions: 3
Actions taken: ['Suck', 'Left', 'Suck']
Correct Outputs!

The test cases test whether the program is working correctly for various positions of the robot and cleanliness of rooms.

Appendix: copy and paste all the source code …

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Apr  5 15:02:12 2022

@author: evastenberg
"""

def test_case():
    states = ['Clean', 'Dirty']                              #defines the room states
    test_results = []                                        #keeps track of results for each case
    for left_state in range(2):                              #for loops to test all 8 cases
        for right_state in range(2):
            for position in range(2):

                room_state = [states[left_state], states[right_state], position]
                action_list = []

                print("Starting state: ", room_state)
                cost = agent_vacuum(room_state, action_list)   #updates room_state, action list
                print("Number of actions: ", cost)             #returns number of actions taken
                print("Actions taken: ", action_list)

                if room_state[0:2] == ['Clean', 'Clean']:      #if the room was cleaned
                    print("Correct Outputs!")
                    test_results.append(cost)
                else:
                    print("Wrong Outputs!")
                    test_results.append(-1)
                print("")


def agent_vacuum(state, actions):
    action_count = 0    #counts actions taken
    while state[0] == 'Dirty' or state[1] == 'Dirty':          #goal state
        run(state, actions)                                    #Runs based on room state and updates
        action_count+=1                                        #counts the actions
    return action_count

def run(state, actions):
    if state[2] == 0:                                          #If agent is in left square and it's dirty, suck
        if state[0] == 'Dirty':
            actions.append('Suck')
            state[0] = 'Clean'                                 #update as clean
        else:
            actions.append('Right')                            #else, left square is clean, move right
            state[2] = 1
    elif state[1] == 'Dirty':                                  #if agent is in right square and right square dirty
        actions.append('Suck')
        state[1] = 'Clean'
    else:                                                      #else agent is in right square, right square clean, move left
        actions.append('Left')
        state[2] = 0


if __name__ == '__main__':
    results = test_case()                                     #calls test case
```