Algorithm for vaccum cleaner for 2 rooms

~~step 1 while status of room 1 and room 2 not equal to clear, repeat steps 2 - 3~~

~~step 2 Go to room 1 and check status~~
~~if status == dirty, begin suck until status = clean~~

~~step 3 Go to room 2 and check status~~
~~if status == dirty, begin suck until status ≠ clea~~

~~step 4 Stop.~~

~~step 1 set status of room 1 and room 2 = dirty~~

~~step 2 go to room 1 and check status,~~
~~if status == clean, set status of room 1 = clea~~

step 1 set varible Flag == 0

step 2 go to room 1 and check status,
if status == clean, set flag = 1
else begin suck and set flag = 1 after clean

step 3 go to room 2 and check status,
if status == clean, set flag = flag + 1
else begin suck and set flag = flag + 1 after clean

step 4 if flag != 2 repeat steps 2 & 3 until flag == 2
else if flag == 2, Stop.

Percept sequence

room 1 Clean	room 2 dirty

① room 1, go to room 2
check

① room 1, go to room 2 ② room 2, clean

① room 1, go to room 2 ② room 2, clean ③ room 2, go to room 1

~~code~~

~~class Vacum Claner:~~
~~def __init__(self%6,~~
~~self. Flag =0~~
~~self.rooms={~~
~~'room_1': False,~~
~~'room_2': False~~
~~}~~

~~def~~

code:

```
class VaccumCleaner Agent:
    def __init__(self, environment):
    self. environment - environment
    self. cleaned_cells = 0
    self. position = (0,0)

    def clean (self):
        while True:
            x,y = self . position

            if self. environment [x][y] = = 'D':
                self. environment[x][y] = 'c'
                self. cleaned_cells += 1
                print (F"cleaned position {self. position}")

        next_position = self . find_next_dirty()
        if next_position :
            print (F"moving to next dirty position{next_po
    else
    print ("No more dirty room found: (leaning comp
    break
```

Date__/__/__

Page____

```
def find_next_dirty (self):
    for i in range(len(self.environmai)):
        for j in range(len(self.enviornmat[i])):
            if self.enviornmat[i][j]=='D':
                return(i,j)
    return None


def display_environment (self):
    for raw in self.environmat:
        print(" ".join(raw))
    print(F"Total cleaned cells: {self.cleaned_cells}")

initial_envionment = [
    ~~['D', 'C', 'D', 'D'],~~   ['D', 'C'],
    ~~['C', 'D', 'C', 'C'],~~
    ~~['D', 'C', 'D', 'C'],~~
    ~~['C', 'C', 'C', 'D']~~
]

agent= VaccumCleanerAgent(initial_environmt)
print("Inital enviromment:")
agent.display_enviornmt

agent.clean()

print("\n Final enviromat")
agent.display_envornmt()
```

8

1/10

Output:

Initial environment:

~~D C D D~~ D C

~~C D C C~~

~~D C D C~~

~~C C C~~ D

Total cleaned cells : 0

cleaned position (0,0)

No more dirty cells found. Cleaning complete

Final environment:

C C

Total cleaned rooms: 1