Experiment #2: Implement a Random Movement Reflex Agent.

Aim/Objective:

Implement a simple reflex agent for a vacuum cleaner.

Description:

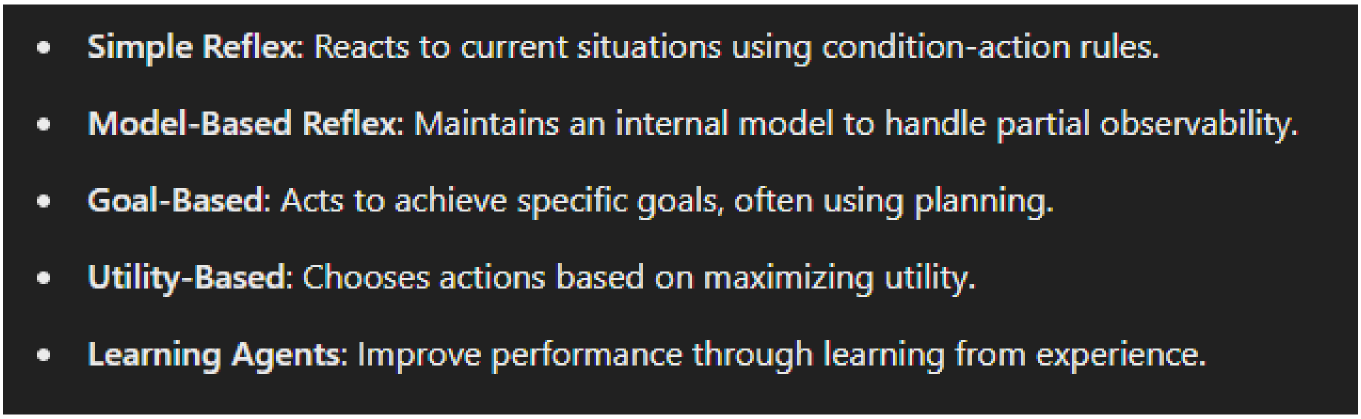
Students will create a simple reflex agent to clean a grid-based environment. The agent will perceive the status of the current cell and decide whether to clean, move, or do nothing.

Pre-Requisites:

The simplicity of the agent's decision-making process makes it an introductory exercise in artificial intelligence, allowing students to understand the concept of reflex agents and their application in autonomous systems.

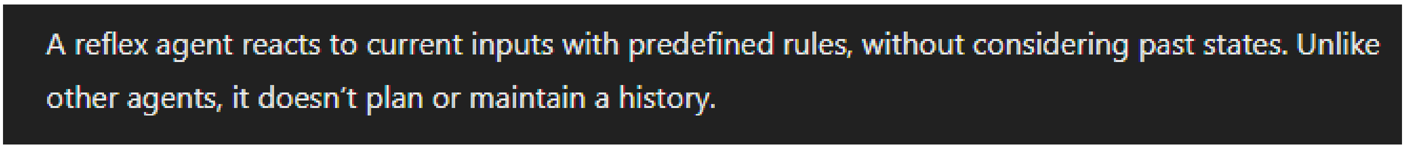
Pre-Lab:

1. What are different types of intelligent agents?



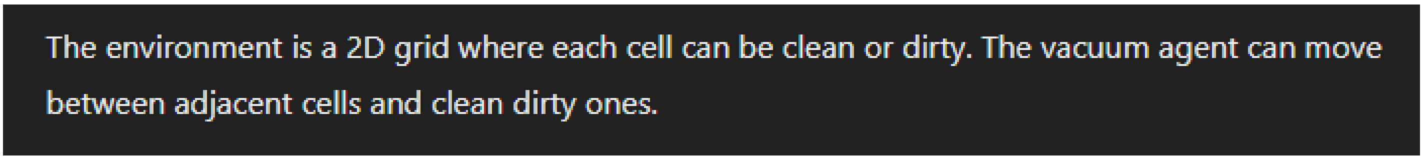
- **Simple Reflex**: Reacts to current situations using condition-action rules.
- **Model-Based Reflex**: Maintains an internal model to handle partial observability.
- **Goal-Based**: Acts to achieve specific goals, often using planning.
- **Utility-Based**: Chooses actions based on maximizing utility.
- **Learning Agents**: Improve performance through learning from experience.

2. What is a reflex agent, and how does it differ from other types of intelligent agents?



A reflex agent reacts to current inputs with predefined rules, without considering past states. Unlike other agents, it doesn’t plan or maintain a history.

3. Describe the grid-based environment in which the vacuum cleaner agent will operate. How is it structured?

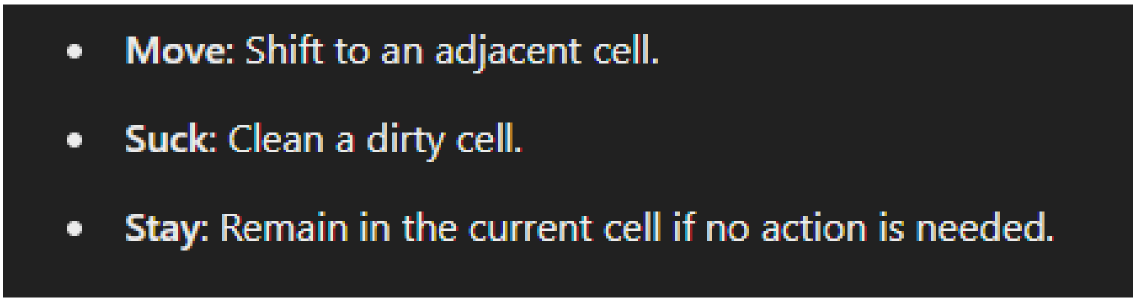


The environment is a 2D grid where each cell can be clean or dirty. The vacuum agent can move between adjacent cells and clean dirty ones.

4. What are the possible states that a cell in the grid-based environment can have? How are these states represented?

Cells can be either **clean** or **dirty**, represented by values like 0 (clean) and 1 (dirty) or using constants.

5. What are the available actions that the vacuum cleaner agent can take in response to the current cell's status?



- **Move:** Shift to an adjacent cell.
- **Suck**: Clean a dirty cell.
- **Stay:** Remain in the current cell if no action is needed.

In-Lab:

An environment consists of a grid of N X N cells. Design an agent that moves randomly within a defined grid environment. The agent can perform the following actions:

- **Move Left:** Move one square to the left (if not at the leftmost edge).
- **Move Right:** Move one square to the right (if not at the rightmost edge).
- **Move Up:** Move one square up (if not at the top edge).
- **Move Down:** Move one square down (if not at the bottom edge).

Write a Python program to create a Random reflex agent that can move in a grid. The agent should be able to sense its current position and then apply any of the random actions as stated above.

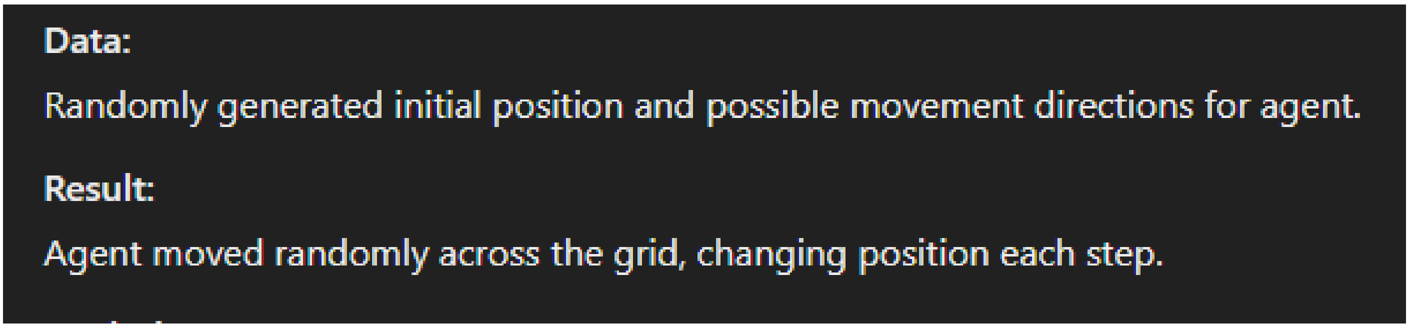
Procedure/Program:

```
import random

class RandomReflexAgent:
    def __init__(self, grid_size):
        self.grid_size = grid_size
        self.position = [
            random.randint(0, grid_size - 1),
            random.randint(0, grid_size - 1),
        ]

    def sense_position(self):
        return tuple(self.position)

    def move(self):
        actions = []
        if self.position[1] > 0:
            actions.append("LEFT")
        if self.position[1] < self.grid_size - 1:
            actions.append("RIGHT")
        if self.position[0] > 0:
            actions.append("UP")
        if self.position[0] < self.grid_size - 1:
            actions.append("DOWN")
        action = random.choice(actions)
        if action == "LEFT":
            self.position[1] -= 1
        elif action == "RIGHT":
            self.position[1] += 1
        elif action == "UP":
            self.position[0] -= 1
        elif action == "DOWN":
            self.position[0] += 1
        return action

def main():
    grid_size = 5
    agent = RandomReflexAgent(grid_size)
    print("Initial Position:", agent.sense_position())
    for step in range(10):
        action = agent.move()
        position = agent.sense_position()
        print(f"Step {step + 1}: Action = {action}, Position = {position}")

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

OUTPUT

```
Initial Position: (0, 1)
Step 1: Action = RIGHT, Position = (0, 2)
Step 2: Action = RIGHT, Position = (0, 3)
Step 3: Action = DOWN, Position = (1, 3)
Step 4: Action = UP, Position = (0, 3)
Step 5: Action = RIGHT, Position = (0, 4)
Step 6: Action = LEFT, Position = (0, 3)
Step 7: Action = LEFT, Position = (0, 2)
Step 8: Action = RIGHT, Position = (0, 3)
Step 9: Action = RIGHT, Position = (0, 4)
Step 10: Action = DOWN, Position = (1, 4)
```

Data and Results:



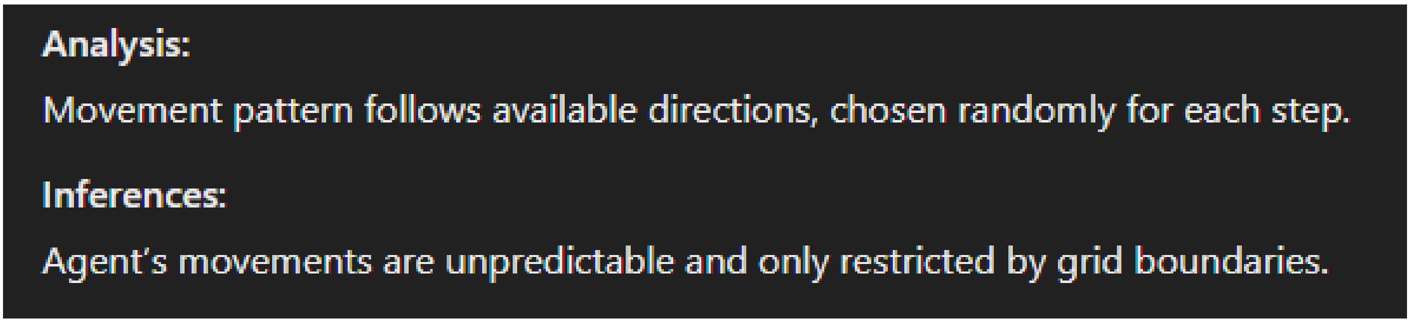
Data:

Randomly generated initial position and possible movement directions for agent.

Result:

Agent moved randomly across the grid, changing position each step.

Analysis and Inferences:



Analysis:

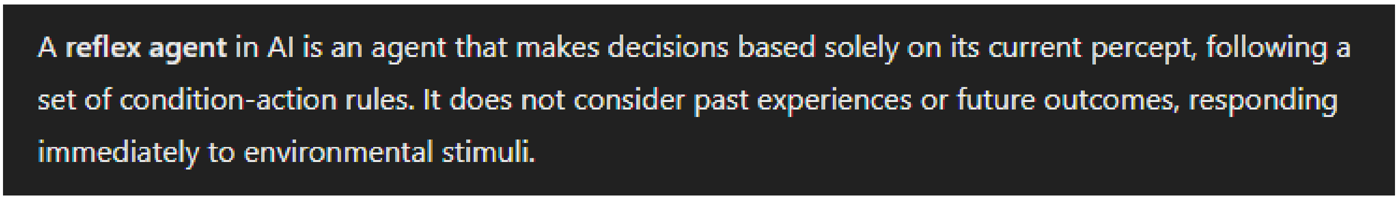
Movement pattern follows available directions, chosen randomly for each step.

Inferences:

Agent's movements are unpredictable and only restricted by grid boundaries.

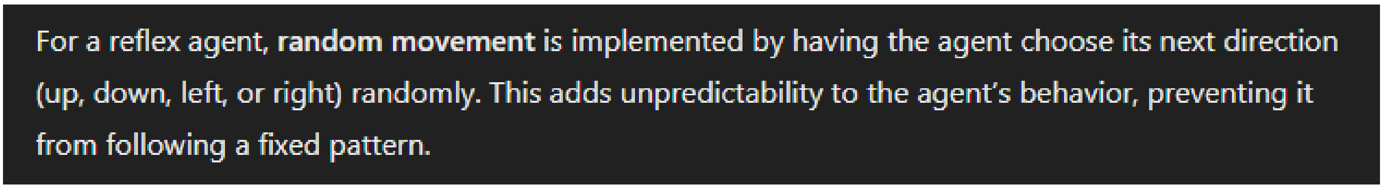
Sample VIVA-VOCE Questions (In-Lab):

1. How would you define a reflex agent in the context of artificial intelligence?



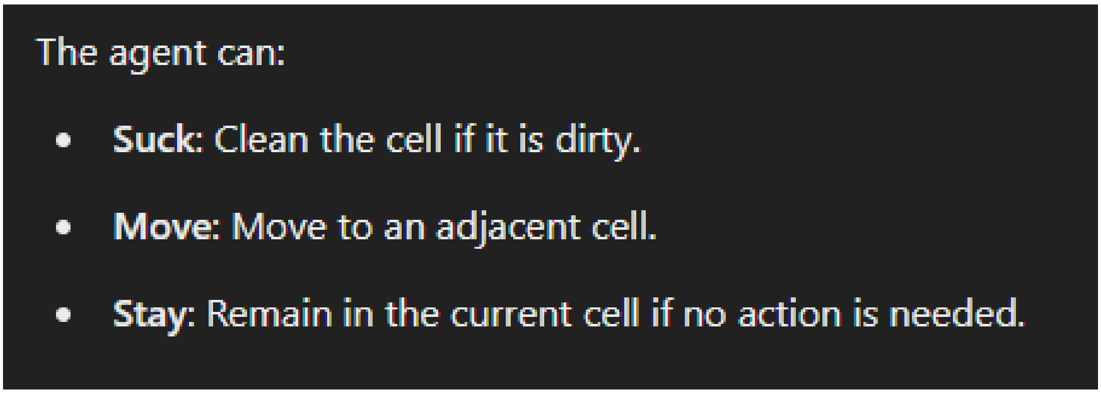
A **reflex agent** in AI is an agent that makes decisions based solely on its current percept, following a set of condition-action rules. It does not consider past experiences or future outcomes, responding immediately to environmental stimuli.

2. Explain the logic behind implementing random movement for a reflex agent.



For a reflex agent, **random movement** is implemented by having the agent choose its next direction (up, down, left, or right) randomly. This adds unpredictability to the agent's behavior, preventing it from following a fixed pattern.

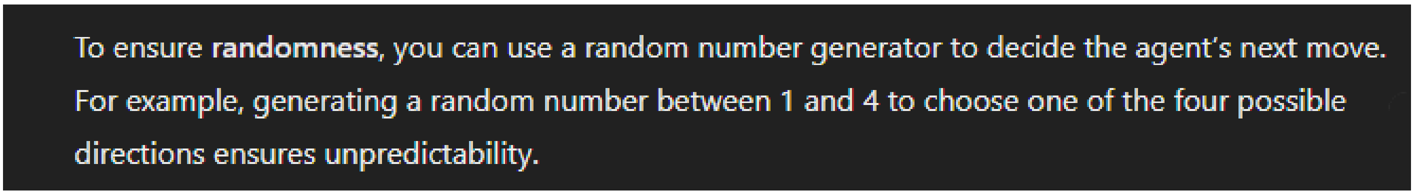
3. What are the possible actions that the agent can take in response to the current cell's status?



The agent can:

- **Suck:** Clean the cell if it is dirty.
- **Move:** Move to an adjacent cell.
- **Stay:** Remain in the current cell if no action is needed.

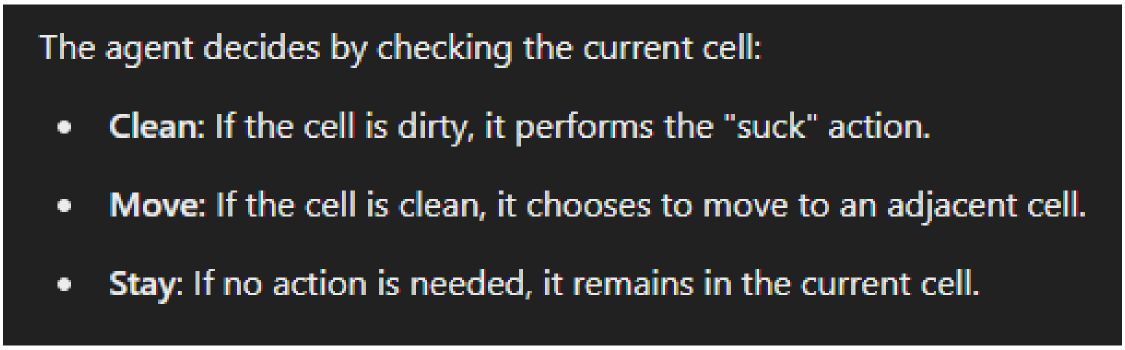
4. How do you ensure that the agent's movements are indeed random and not following a predictable pattern?



To ensure **randomness**, you can use a random number generator to decide the agent's next move. For example, generating a random number between 1 and 4 to choose one of the four possible directions ensures unpredictability.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	[@KLWKS_BOT] THANOS

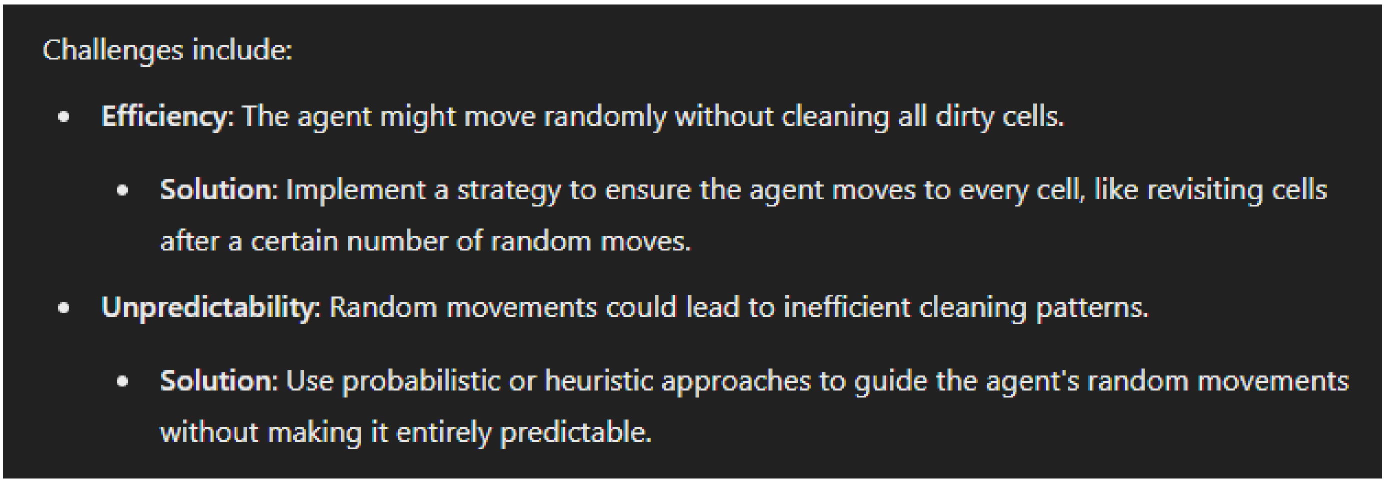
5. How does the agent decide whether to clean, move, or do nothing based on the current cell's status?



The agent decides by checking the current cell:

- **Clean:** If the cell is dirty, it performs the "suck" action.
- **Move:** If the cell is clean, it chooses to move to an adjacent cell.
- **Stay**: If no action is needed, it remains in the current cell.

6. What are the potential challenges you might face when implementing a random movement reflex agent, and how would you address them?



Challenges include:

- **Efficiency**: The agent might move randomly without cleaning all dirty cells.
  - **Solution**: Implement a strategy to ensure the agent moves to every cell, like revisiting cells after a certain number of random moves.
- **Unpredictability:** Random movements could lead to inefficient cleaning patterns.
  - **Solution**: Use probabilistic or heuristic approaches to guide the agent's random movements without making it entirely predictable.

Post-Lab:

create a vacuum cleaner agent with two grid of squares (A and B) environment.

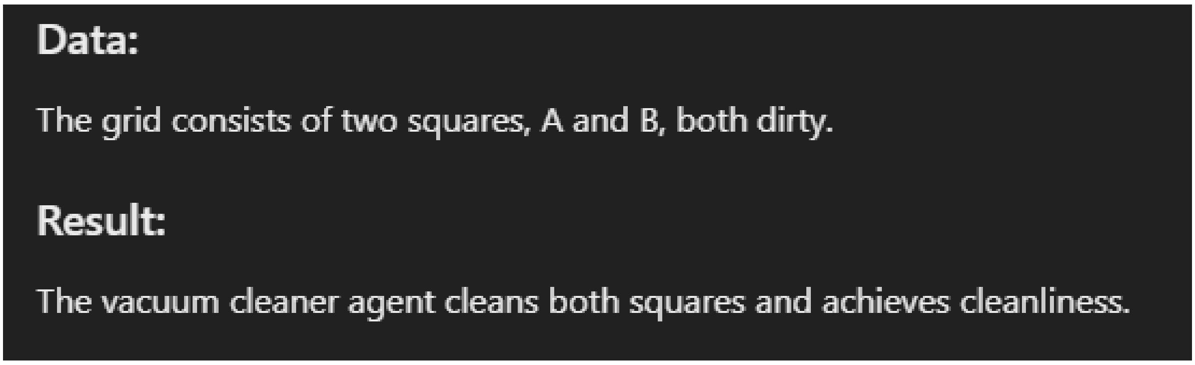
a) **Procedure/Program:**

```
class VacuumCleanerAgent:
    def __init__(self, grid):
        self.grid = grid
        self.position = 'A'

    def is_dirty(self):
        return self.grid[self.position] == 'dirty'

    def clean(self):
        print(f"Cleaning {self.position}...")
        self.grid[self.position] = 'clean'

    def move(self):
        if self.position == 'A':
            self.position = 'B'
        elif self.position == 'B':
            self.position = 'A'

    def perform_action(self):
        if self.is_dirty():
            self.clean()
        else:
            self.move()

    def run(self):
        while 'dirty' in self.grid.values():
            self.perform_action()
        print("Environment cleaned!")
        print("Final grid state:", self.grid)

grid = {'A': 'dirty', 'B': 'dirty'}
agent = VacuumCleanerAgent(grid)
agent.run()
```

OUTPUT

```
Cleaning A...
Cleaning B...
Environment cleaned!
Final grid state: {'A': 'clean', 'B': 'clean'}
```

b) Data and Results:



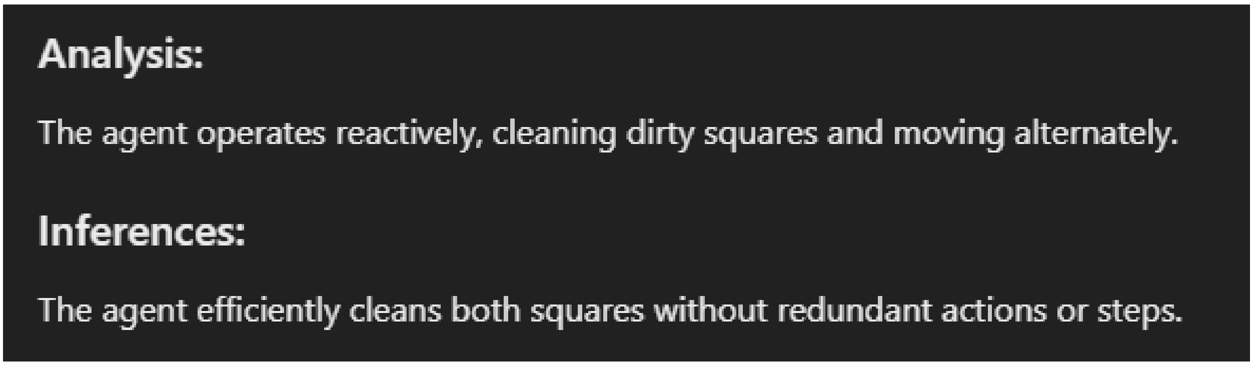
Data:

The grid consists of two squares, A and B, both dirty.

Result:

The vacuum cleaner agent cleans both squares and achieves cleanliness.

c) Analysis and Inferences:



Analysis:

The agent operates reactively, cleaning dirty squares and moving alternately.

Inferences:

The agent efficiently cleans both squares without redundant actions or steps.

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date