Implement vacuum cleaner agent

```
def vacuum_world():
    # Initializing goal_state
    # 0 indicates Clean and 1 indicates Dirty
    goal_state = {'A': '0', 'B': '0'}
    cost = 0
    location_input = input("Enter Location of Vacuum: ")  # User input for location vacuum is placed
    status_input = input("Enter status of " + location_input + " (0 for Clean, 1 for Dirty): ")  # User input if location is dirty or clean
    status_input_complement = input("Enter status of other room (0 for Clean, 1 for Dirty): ")

    print("Initial Location Condition: " + str(goal_state))

    if location_input == 'A':
        # Location A is Dirty.
        print("Vacuum is placed in Location A")
        if status_input == '1':
            print("Location A is Dirty.")
            # Suck the dirt and mark it as clean
            goal_state['A'] = '0'
            cost += 1  # Cost for suck
            print("Cost for CLEANING A: " + str(cost))
            print("Location A has been Cleaned.")

            if status_input_complement == '1':
                # If B is Dirty
                print("Location B is Dirty.")
                print("Moving right to Location B.")
                cost += 1  # Cost for moving right
                print("COST for moving RIGHT: " + str(cost))
                # Suck the dirt and mark it as clean
```

```
            goal_state['B'] = '0'
            cost += 1  # Cost for suck
            print("COST for SUCK: " + str(cost))
            print("Location B has been Cleaned.")
        else:
            print("No action needed; Location B is already clean.")
    else:
        print("Location A is already clean.")
        if status_input_complement == '1':  # If B is Dirty
            print("Location B is Dirty.")
            print("Moving RIGHT to Location B.")
            cost += 1  # Cost for moving right
            print("COST for moving RIGHT: " + str(cost))
            # Suck the dirt and mark it as clean
            goal_state['B'] = '0'
            cost += 1  # Cost for suck
            print("COST for SUCK: " + str(cost))
            print("Location B has been Cleaned.")
        else:
            print("No action needed; Location B is already clean.")

else:  # Vacuum is placed in location B
    print("Vacuum is placed in Location B")
    if status_input == '1':
        print("Location B is Dirty.")
        # Suck the dirt and mark it as clean
        goal_state['B'] = '0'
        cost += 1  # Cost for suck
        print("COST for CLEANING B: " + str(cost))
        print("Location B has been Cleaned.")
```

```
        if status_input_complement == '1':  # If A is Dirty
            print("Location A is Dirty.")
            print("Moving LEFT to Location A.")
            cost += 1  # Cost for moving left
            print("COST for moving LEFT: " + str(cost))
            # Suck the dirt and mark it as clean
            goal_state['A'] = '0'
            cost += 1  # Cost for suck
            print("COST for SUCK: " + str(cost))
            print("Location A has been Cleaned.")
        else:
            print("No action needed; Location A is already clean.")
    else:
        print("Location B is already clean.")
        if status_input_complement == '1':  # If A is Dirty
            print("Location A is Dirty.")
            print("Moving LEFT to Location A.")
            cost += 1  # Cost for moving left
            print("COST for moving LEFT: " + str(cost))
            # Suck the dirt and mark it as clean
            goal_state['A'] = '0'
            cost += 1  # Cost for suck
            print("COST for SUCK: " + str(cost))
            print("Location A has been Cleaned.")
        else:
            print("No action needed; Location A is already clean.")

# Done cleaning
print("GOAL STATE: ")
print(goal_state)
print("Performance Measurement: " + str(cost))
```

```
# Call the function to run the vacuum world simulation
vacuum_world()
```

OUTPUT

Output

```
Enter Location of Vacuum: B
Enter status of B (0 for Clean, 1 for Dirty): 1
Enter status of other room (0 for Clean, 1 for Dirty): 1
Initial Location Condition: {'A': '0', 'B': '0'}
Vacuum is placed in Location B
Location B is Dirty.
COST for CLEANING B: 1
Location B has been Cleaned.
Location A is Dirty.
Moving LEFT to Location A.
COST for moving LEFT: 2
COST for SUCK: 3
Location A has been Cleaned.
GOAL STATE:
{'A': '0', 'B': '0'}
Performance Measurement: 3

=== Code Execution Successful ===
```