Lab #09

Name: Maryum Shakeel

Sap ID: 48406

Subject: AI Lab

Batch: BSCS 6th Semester

Submitted by: Miss Ayesha Akram

Task 1

Solution:

```
import random


# Step 1: Get number of players
n = int(input("Enter number of players: "))


# Step 2: Create players and cards list
players = ["Player " + str(i+1) for i in range(n)]
suits = ["Spades", "Hearts", "Diamonds", "Clubs"]
cards = []


# Create n cards with random suits and values (1 to 13)
for i in range(n):
    value = random.randint(1, 13)
    suit = random.choice(suits)
    cards.append((value, suit))


# Step 3: Keep track of assigned players and cards
assigned = {}
```

```
# Step 4: Roll dice to assign
while len(assigned) < n:
    player_index = random.randint(0, n-1)
    card_index = random.randint(0, n-1)

    if player_index not in assigned and card_index not in assigned.values():
        assigned[player_index] = card_index
        print(f"{players[player_index]} got card {cards[card_index]}")

# Step 5: Decide winner (value first, suit second)
suit_rank = {"Clubs": 1, "Diamonds": 2, "Hearts": 3, "Spades": 4}
winner = None

for player_index, card_index in assigned.items():
    value, suit = cards[card_index]
    if not winner:
        winner = (player_index, value, suit)
    else:
        if value > winner[1] or (value == winner[1] and suit_rank[suit] >
suit_rank[winner[2]]):
            winner = (player_index, value, suit)
```

```
# Step 6: Print winner

print(f"\nWinner is {players[winner[0]]} with card ({winner[1]}, {winner[2]})")
```

Output

```
"C:\Users\CS COMPUTERS\PyCharmMiscProject\.venv\
Enter number of players: 3
Player 2 got card (13, 'Clubs')
Player 1 got card (6, 'Spades')
Player 3 got card (11, 'Hearts')

 Winner is Player 2 with card (13, Clubs)

Process finished with exit code 0
```

Task 2

Solution:

```
# Goal-based Agent

class GoalBasedAgent:

    def __init__(self, goal):

        self.goal = goal


    def perform_task(self):

        print(f"Goal-based agent: Performing task to achieve goal: {self.goal}")


# Model-based Agent
```

```
class ModelBasedAgent:
    def __init__(self, environment_state):
        self.state = environment_state

    def perform_task(self):
        if self.state == "dirty":
            print("Model-based agent: Cleaning the room...")
        else:
            print("Model-based agent: Room is clean!")

# Utility-based Agent
class UtilityBasedAgent:
    def __init__(self, options):
        self.options = options

    def perform_task(self):
        best_option = max(self.options, key=self.options.get)  # Selects the most
useful option
        print(f"Utility-based agent: Choosing best option: {best_option}")

# Test all agents
print("Testing Goal-based Agent:")
```

```
goal_agent = GoalBasedAgent("Reach the destination")
goal_agent.perform_task()


print("\nTesting Model-based Agent:")
model_agent = ModelBasedAgent("dirty")
model_agent.perform_task()


print("\nTesting Utility-based Agent:")
preferences = {"Laptop": 9, "Phone": 7, "Tablet": 8}
utility_agent = UtilityBasedAgent(preferences)
utility_agent.perform_task()
```

Output

```
"C:\Users\CS COMPUTERS\PyCharmMiscProject\.venv\Scripts\python.exe" "C:\Users\
Testing Goal-based Agent:
Goal-based agent: Performing task to achieve goal: Reach the destination

Testing Model-based Agent:
Model-based agent: Cleaning the room...

Testing Utility-based Agent:
Utility-based agent: Choosing best option: Laptop

Process finished with exit code 0
```