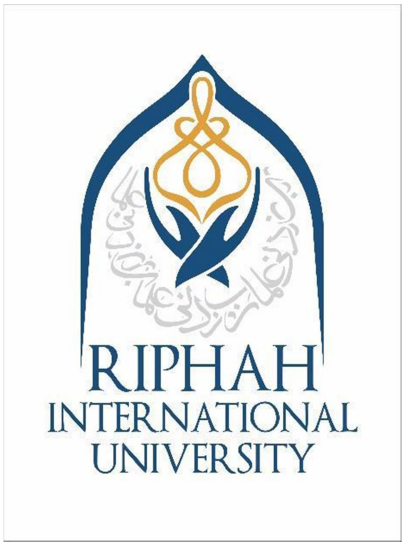
LAB # 09

Artificial Intelligence



Name : Adeesha Sherani

Sap Id : 47379

Submitted to : Mam Ayesha

Task 01:

```python
import random

class CasinoAgent:  1 usage
    def __init__(self, n):
        self.players = list(range(1, n + 1))
        self.cards = self.create_cards(n)
        self.assigned = {}

    def create_cards(self, n):  1 usage
        suits = ['Spades', 'Hearts', 'Diamonds', 'Clubs']
        cards = []
        for i in range(n):
            number = random.randint( a: 1,  b: 13)
            suit = suits[i % 4]
            cards.append((number, suit))
        return cards

    def roll_dice(self):  1 usage
        return random.choice(self.players), random.choice(range(1, len(self.cards) + 1))

    def assign_cards(self):  1 usage
        while len(self.assigned) < len(self.players):
            player, card_num = self.roll_dice()
            if player not in self.assigned and card_num not in [v[0] for v in self.assigned.values()]:
                self.assigned[player] = (card_num, self.cards[card_num - 1])
                print(f"Player {player} got card {self.cards[card_num - 1]}")

    def announce_winner(self):  1 usage
        def card_rank(card):
            suit_order = {'Spades': 4, 'Hearts': 3, 'Diamonds': 2, 'Clubs': 1}
            number, suit = card
            return number * 10 + suit_order[suit]

        winner = max(self.assigned.items(), key=lambda item: card_rank(item[1][1]))
        print(f"\nWinner is Player {winner[0]} with card {winner[1][1]}")

agent = CasinoAgent(n=4)
agent.assign_cards()
agent.announce_winner()
```

Output:

```
Player 2 got card (10, 'Spades')
Player 1 got card (10, 'Diamonds')
Player 3 got card (6, 'Hearts')
Player 4 got card (7, 'Clubs')

Winner is Player 2 with card (10, 'Spades')

Process finished with exit code 0
```

Task 02:

```python
class GoalAgent:  1 usage
    def __init__(self):
        self.location = 0
        self.goal = 5

    def act(self):  1 usage
        while self.location < self.goal:
            print(f"Moving from {self.location} to {self.location + 1}")
            self.location += 1
        print("Goal Reached!")


class ModelAgent:  1 usage
    def __init__(self):
        self.history = []
        self.state = 'Start'

    def perceive(self, info):  2 usages
        self.history.append(info)
        self.state = info

    def act(self):  2 usages
        if self.state == 'Dirty':
            print("Cleaning...")
        else:
            print("Nothing to do.")
```

```
class UtilityAgent:  1 usage
    def __init__(self, options):
        self.options = options

    def decide(self):  1 usage
        best = max(self.options, key=lambda x: x['utility'])
        print(f"Best option: {best['name']} with utility {best['utility']}")


print("=== Goal Agent ===")
goal = GoalAgent()
goal.act()

print("\n=== Model Agent ===")
model = ModelAgent()
model.perceive('Dirty')
model.act()
model.perceive('Clean')
model.act()

print("\n=== Utility Agent ===")
options = [
    {'name': 'Option A', 'utility': 5},
    {'name': 'Option B', 'utility': 7},
    {'name': 'Option C', 'utility': 6}
]
```

```
utility = UtilityAgent(options)
utility.decide()
```

Output:

```
=== Goal Agent ===
Moving from 0 to 1
Moving from 1 to 2
Moving from 2 to 3
Moving from 3 to 4
Moving from 4 to 5
Goal Reached!

=== Model Agent ===
Cleaning...
Nothing to do.

=== Utility Agent ===
Best option: Option B with utility 7
```