Use Case #1

Finding the Winning Strategy in a Card Game using Python

PROBLEM STATEMENT

In a two-player card game, each player draws a card from a deck. The player with the higher card value wins the round.
Your goal is to find a winning strategy — that is, determine the probability of winning or the best move for each round based on the cards drawn.

AIM:

To implement a Python program that analyses possible moves in a card game and determines the optimal strategy (or the player most likely to win).

PROBLEM UNDERSTANDING:

➢ **Input:**

- Two lists (or arrays) of integers:
  - Player A's cards
  - Player B's cards
- Each list contains the same number of cards (say, *n*).

➢ **Goal:**

- Simulate the game and determine the best possible outcome for Player A.
- Find:
  1. The number of rounds won by each player.
  2. The final winner (Player A or Player B).
  3. Optionally, the best order (strategy) of cards that gives the highest score.

➢ **Constraints**

1. Both players have the same number of cards.
2. Each card can be used only once in the game.
3. Card values are positive integers.
4. If both players play cards of the same value, the round is a draw (no points).
5. The game ends when all cards have been played once.

ALGORITHM:

Step 1: Start the program.
Step 2: Input two lists — one for Player A's cards and another for Player B's cards.
Step 3: Initialize two variables score_A = 0 and score_B = 0.
Step 4: Loop through both lists:
- **a.** Compare the cards at the same index.
- **b.** If Player A's card > Player B's card → increment score_A.
- **c.** If Player B's card > Player A's card → increment score_B.
- **d.** Otherwise, it's a draw (no points).

Step 5: After all rounds, compare total scores.
- **a.** If score_A > score_B → Player A wins.
- **b.** If score_B > score_A → Player B wins.
- **c.** Else → It's a draw.

Step 6: Display the total scores and the winner.
Step 7: Stop the program.

SOURCE CODE:

```
# Sample card game: highest total value wins
def analyze_game(players_cards):
    scores = {}
    for player, cards in players_cards.items():
        scores[player] = sum(cards)

    # Determine winner
    winner = max(scores, key=scores.get)
    print("Player Scores:")
    for player, score in scores.items():
        print(f"{player}: {score}")
    print(f"\nMost likely to win: {winner} with score {scores[winner]}")

# Example input: dictionary of players and their card values
players_cards = {
    "Alice": [10, 5, 7],
    "Bob": [8, 9, 6],
    "Charlie": [4, 10, 10]
}

analyze_game(players_cards)
```

INPUT AND OUTPUT:

Output

```
Player Scores:
Alice: 22
Bob: 23
Charlie: 24

Most likely to win: Charlie with score 24

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

RESULT:

To implement a Python program that analyses possible moves in a card game and determines the optimal strategy (or the player most likely to win) is successfully implemented.