**Background and Questions**

Background of the Problem

SET is a card game based on pattern recognition. The goal is to find a "set" of three cards that follow specific rules. Each card has four attributes: number, shape, color and shading, with three possible values for each.

A valid set consists of three cards where, for each attribute, the values are either all the same or all different. Mathematically, a SET card can be represented as a vector with four values in {0,1,2}. A set is a group of three such vectors where each attribute follows the set rules.

In this project, we developed a Python program where a player competes against a computer in SET. There are always 12 cards on the table. The first to find a set scores a point. If the player does not find a set within a time limit, the computer gets a point and immediately reveals the set.

**Key Questions and Challenges**

1**. How to Build the Game?**

- How should the game be structured using classes and functions?

- What data structures are best for handling the deck, table, and found sets?

- How can we efficiently check if a set exists among the displayed cards?

- What is pygame, and how do we use it?

**2. How to Play Against the Computer?**

- What strategy should the computer use to find a set immeditely?

- How do we ensure fair gameplay while making the computer challenging?

- How can the computer find a set instantly but reveal it only after the player runs out of time?

-

**3. How to Represent the Cards in Code?**

- What class structure should represent a card?

- How can we efficiently check if three selected cards form a valid set?

- How should the deck be shuffled and drawn from?

**4. How to Build the Game Interface?**

- How should the cards be displayed to the player?

- How can the player select cards?

- How should we give visual feedback when a set is found or a wrong selection is made?

**5. How to Handle Timing and Scoring?**

- How should we enforce a time limit for the player?

- How should the score be updated and displayed?

- How should the computer reveal the set only after the player’s time runs out?

**6. How to Implement Buttons and User Interaction?**

- How should buttons be created for selecting cards and submitting a set?

- What visual indicators should show button states (e.g., selected, disabled)?

- How should error messages or game instructions be displayed to the player?

**7. How to Optimize Performance?**

- What is the most efficient way to check for sets among 12 cards?

- How can we reduce computational overhead while keeping the game fair?

- What are the space and time complexities of our approach?

**8. How to Ensure Random and Fair Card Distribution?**

- How should the deck be shuffled and drawn from?

- How can we ensure randomness while avoiding infinite loops when no sets exist?

- How should we handle cases where extra cards need to be added?

By answering these questions, we can create a well-structured and efficient SET game in Python, offering an engaging and fair experience against a computer opponent.