**Assignment Part 2**

**Group Information**



**Problem statement 1: Gaming**

**Catch-Up with numbers**

**Introduction**

The Catch-Up Strategy Game is a two-player turn-based game where each player selects numbers from a predefined set in order to maximize their total score. The game incorporates elements of strategy and decision-making, making it an engaging mathematical challenge.

**Game Rules**

1. The game begins with numbers ranging from 1 to N, where N is decided by the user.
2. Player 1 (P1) starts by selecting a single number from the set.
3. Thereafter, players take turns choosing one or more numbers such that their sum is equal to or greater than the opponent’s last sum.
4. The game continues until all numbers have been selected.
5. The player with the highest sum at the end wins, or the game results in a tie if both players have equal sums.

**Game Implementation**

The game is implemented using Python, utilizing key programming concepts such as:

* Minimax Algorithm for optimal move selection.
* Greedy Subset Selection for strategy determination.
* Randomization for variable AI difficulty settings.

**Algorithm Details**

**1. Minimax Decision-Making**

* The minimax function evaluates the possible moves by simulating future game states.
* The AI tries to maximize its advantage while assuming the opponent plays optimally.
* The function operates with a defined depth limit to balance efficiency and performance.

**2. Subset Selection Strategy**

* The function find\_optimal\_subset(numbers, threshold) selects the smallest subset of numbers whose sum is at least the required threshold.
* This ensures the AI meets the game’s constraint of exceeding or matching the opponent’s last sum.

**3. AI Difficulty Levels**

The AI difficulty is set based on user input:

* Easy Mode: AI selects random numbers from the available set.
* Medium Mode: AI uses greedy subset selection to maximize immediate gains.
* Hard Mode: AI employs Minimax strategy to evaluate moves deeply and optimize long-term gains.

**User Interaction**

1. The game starts with user input specifying the range of numbers (N).
2. The user selects numbers using comma-separated inputs.
3. The AI processes its moves based on the chosen difficulty.
4. The game continues until all numbers are exhausted.

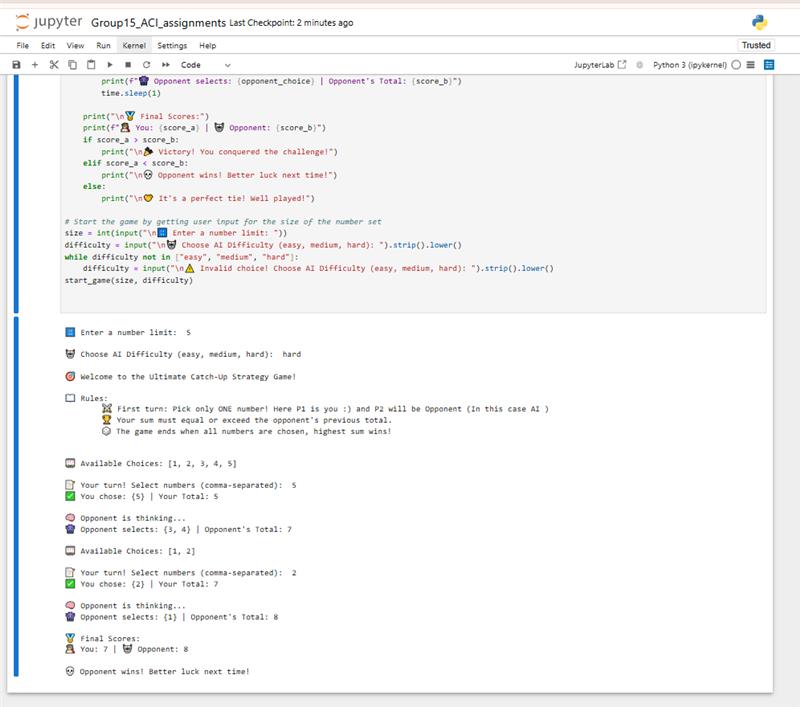
**Edge Cases Considered**

* Invalid Inputs Handling: Ensuring the user picks numbers from the available set.
* First Turn Constraint: Restricting P1 to pick only one number initially.
* End-Game Scenarios: Handling situations where no valid moves are left.

**Conclusion**

The Catch-Up Strategy Game integrates mathematical decision-making and AI strategy into an interactive experience. The use of Minimax, Greedy Selection, and Difficulty Scaling provides a dynamic challenge for players of all skill levels.

**Output Samples:**



**Problem statement 2: Logic**

Prolog Decision Tree-Based Classifier Documentation

**Introduction**

This Prolog program implements a decision tree-based classifier that predicts a class (c0 or c1) based on user inputs. The decision tree is structured with binary attributes (true/false), and classification is done by traversing the tree according to user responses.

**Program Structure**

The program consists of the following key components:

* predict\_class/1: The main predicate that initiates classification.
* Decision rules: A set of predicates that guide the classification based on user inputs.
* User input handling (ask/2): Asks users for attribute values.
* Start predicate (start/0): Begins the classification process.

**How the Classifier Works**

1. The classifier starts by asking about attribute a5.
2. Based on the response, it traverses the decision tree using a series of decision predicates.
3. Each decision predicate checks an attribute and either:
   * Determines a class (c0 or c1).
   * Asks for the next relevant attribute.
4. Once a class is determined, it is displayed to the user.

**Decision Rules**

The classifier follows a structured decision tree:

1. If a5 is false, classify as c0.
2. If a5 is true, ask for a8:
   * If a8 is false, ask for a9:
     + If a9 is false, ask for a2:
       - If a2 is false, classify as c0.
       - If a2 is true, ask for a0:
         * If a0 is false, ask for a4:

If a4 is false, classify as c1.

If a4 is true, classify as c0.

* + - * + If a0 is true, classify as c1.
    - If a9 is true, classify as c1.
  + If a8 is true, ask for a1:
    - If a1 is false, ask for a2:
      * If a2 is false, ask for a0:
        + If a0 is false, classify as c1.
        + If a0 is true, classify as c0.
      * If a2 is true, ask for a4:
        + If a4 is false, classify as c1.
        + If a4 is true, classify as c0.
    - If a1 is true, classify as c0.

Key Predicates Explained

predict\_class/1

* Entry point for classification.
* Calls ask/2 to get the first attribute (a5).
* Calls decide/2 to start decision-making.

decide/2

* Determines the classification based on a5.
* Calls further decision predicates (decide\_a8, decide\_a9, decide\_a1, etc.).

ask/2

* Asks the user to provide a true/false value for a given attribute.
* Uses write/1 to prompt input and read/1 to capture response.

start/0

* Starts the classification process by calling predict\_class/1.
* Displays the final classification result.

**Example Interaction**

?- start.  
Enter value for a5 (true/false): true.  
Enter value for a8 (true/false): false.  
Enter value for a9 (true/false): false.  
Enter value for a2 (true/false): true.  
Enter value for a0 (true/false): false.  
Enter value for a4 (true/false): true.  
The predicted class is: c0.

**Features and Benefits**

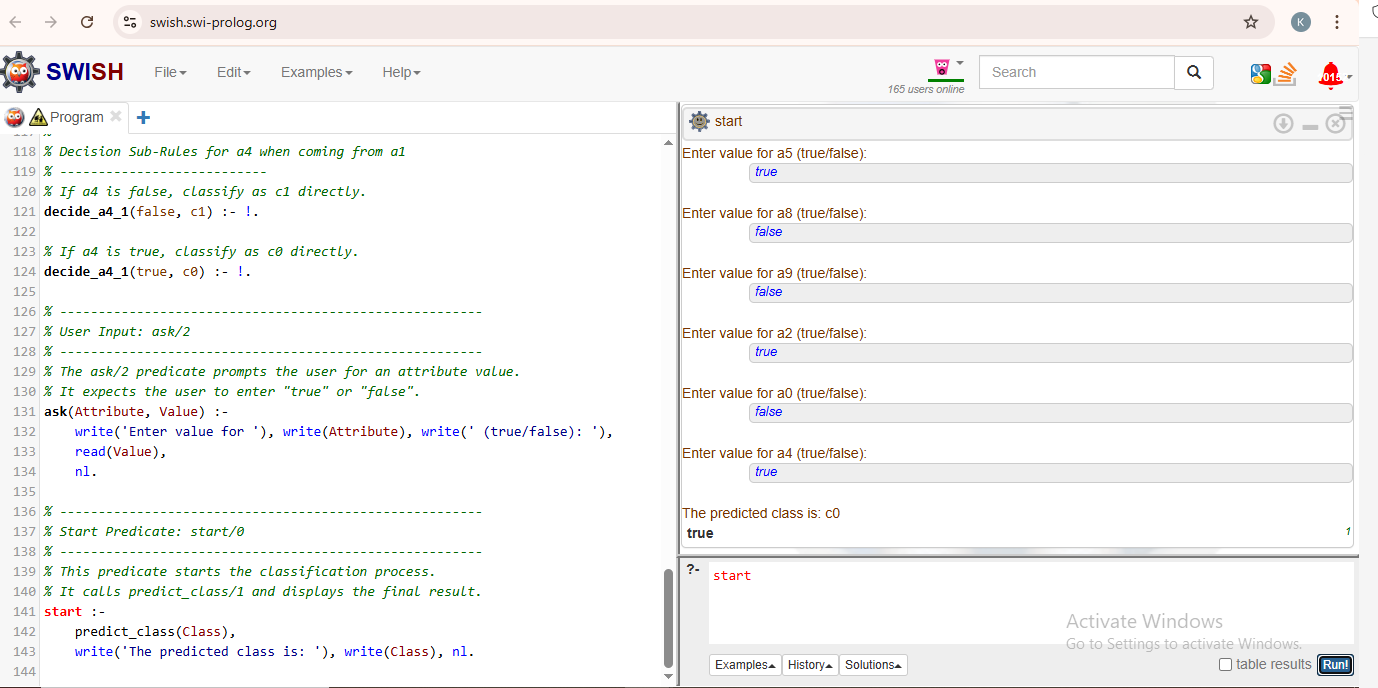
* Binary Decision Tree: Efficient classification using simple true/false questions.
* Interactive Input: Prompts user dynamically based on prior responses.
* Deterministic Rules: Ensures reproducible results based on decision structure.
* Prolog-Based AI: Demonstrates logical reasoning in classification.

**Conclusion**

This Prolog-based classifier demonstrates a simple yet effective decision tree model. It allows interactive classification based on binary attribute values and showcases logical AI decision-making.

**Sample Outputs:**

?- start.  
Enter value for a5 (true/false): true.  
Enter value for a8 (true/false): false.  
Enter value for a9 (true/false): false.  
Enter value for a2 (true/false): true.  
Enter value for a0 (true/false): false.  
Enter value for a4 (true/false): true.  
The predicted class is: c0.



* **When a5 is false**

