Assignment 1: Policy Iteration in the Repeated Prisoner's Dilemma

Objective

In this assignment, you will bridge Game Theory and Reinforcement Learning. You will build a custom environment using the **Gymnasium API** to model the Repeated Prisoner's Dilemma (RPD). You will explicitly define the underlying Markov Decision Process (MDP) and use **Policy Iteration** to compute the optimal strategy against opponents with varying personalities and reliability.

---

1. The Game Mechanics

You and an opponent simultaneously choose an action: **Cooperate (C)** or **Defect (D)**.

The Payoff Matrix:

- R (Reward): You Cooperate, Opponent Cooperates.
- S (Sucker): You Cooperate, Opponent Defects.
- T (Temptation): You Defect, Opponent Cooperates.
- P (Punishment): You Defect, Opponent Defects.

You \ Opponent	**Cooperate (C)**	**Defect (D)**
Cooperate (C)	R = 3	S = 0
Defect (D)	T = 5	P = 1

The game repeats indefinitely with a discount factor $\gamma \in (0,1)$.

---

2. Part I: Build the Environment (Implementation)

Create a Python class that inherits from `gymnasium.Env`. Your environment must support different "Opponent Strategies" and "State Representations".

A. Opponent Strategies

Your agent will play against four specific opponent strategies:

1. **ALL-C:** Always Cooperates.
2. **ALL-D:** Always Defects.
3. **Tit-for-Tat (TFT):** Deterministic. Starts with C, then copies whatever action you took in the previous round ($b_t = a_{t-1}$).
4. **Imperfect Tit-for-Tat:** Stochastic.
    - **90% chance:** Copies your previous move.
    - **10% chance:** It “slips” and does the opposite of your previous move.
    - *Note: This introduces stochasticity into your transition matrix.*

B. Observation Schemes (State Definitions)

How much history does your agent see? You must implement these two variations:

1. **Memory-1:** You see the outcome of the previous round (a_{t-1}, b_{t-1}).
2. **Memory-2:** You see your last two moves and the opponent's last two moves ($a_{t-1}, a_{t-2}, b_{t-1}, b_{t-2}$).

Initial State Handling: Since the state depends on history, we must define the "history" at the start of the episode ($t = 0$). Assume that prior to the game starting, both agents Cooperated.

- **Memory-1 Start State:** (C, C)
- **Memory-2 Start State:** (C, C, C, C)

---

3. Part II: Define the MDP (The Math)

Reinforcement Learning relies on the Markov Property. For each Observation Scheme defined above, you must mathematically define the "world" and the Transition Matrix.

Analysis Requirement:

For each observation scheme and opponent, calculate and report:

1. **The State Space:** List the unique states and the total count.
2. **Transition Probabilities** P(s'|s,a): Define the probability of moving to the state s' given the current state and action.
3. **Rewards** R(s,a): The expected immediate reward.

> **Hint:** Pay close attention to the *Imperfect TFT* opponent. Your Transition Matrix and expected reward must account for the stochastic opponent.

4. Part III: Policy Iteration (The Algorithm)

Implement **Tabular Policy Iteration** from scratch (do not use a library like standard solvers). You should implement the algorithm as learned in class (iterating between Policy Evaluation and Policy Improvement).

---

5. Part IV: Experiments & Analysis

Run experiments varying the Discount Factor $\gamma \in \{0.1, 0.5, 0.9, 0.99\}$ and the Opponent type.

Simulation verification: After finding the optimal policy, play the game for 50 episodes (each 50 steps long). Calculate and report the average cumulative reward to validate your theoretical results against the actual environment.

Key Questions to Answer:

1. **The Discount Factor:** Does a low γ (shortsightedness) force you to Defect? At what γ value does Cooperation become optimal against TFT?
2. **Memory Depth:** Compare Memory-1 vs. Memory-2.
    - Does remembering two steps back allow for a higher reward against any of the opponents? Why or why not?
    - Propose a hypothetical opponent strategy (which relies on history) where a Memory-2 agent *would* strictly outperform a Memory-1 agent. Explain why the extra history is necessary.
3. **Noise Analysis:** Compare the optimal policy against **TFT** vs. **Imperfect TFT**.
    - Does the 10% noise break the cooperation?
    - Does the agent learn to be "forgiving" or does it revert to "always defect"?

---

6. Deliverables & Submission Instructions

Submission is in **pairs**. Submit exactly **two files** via the portal.

1. The Report (PDF)

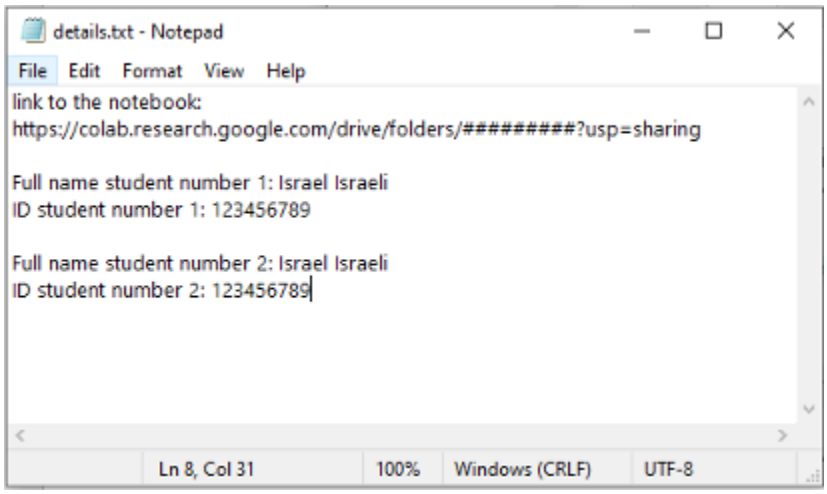
Include the following analysis and results:

- **MDP Definition (Part II):** Report the State Space, Reward Function, and Transition Tables for each scenario.
- **Analysis (Part IV):** Answers to all questions, including convergence plots, simulation verification results (average reward over 50 episodes), and an executive summary.

2. Submission Details (.txt)

A text file containing your code link and student details.

- **Google Colab:**
  - Run all cells before saving. **Outputs must be visible** without running the code.
  - **Permissions:** Set to "Anyone with the link."
- **File Content:** Copy the format below exactly.



To sum up, in the submission box, submit the following files:

1. **details.txt**
2. **report_ID1_ID2.pdf**