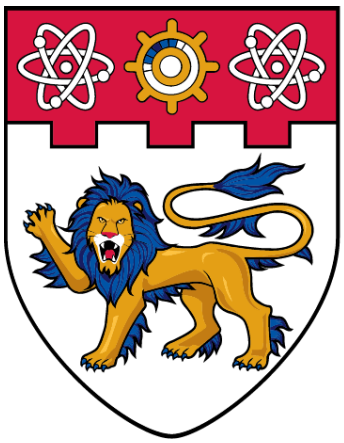

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

CZ4046: Intelligent Agent

Dongwoo Han

U1722048E

Assignment II

Approach and Thought process

Prisoner's Dilemma is a classical **non-zero-sum** game in Game Theory.

In 1983, Robert Axelrod held a tournament where 14 different groups have participated. This tournament was an iterated 2-prisoner's dilemma. The winner of this tournament was TitForTat and Axelrod has come up with the following characteristics for high performing strategies.

1. Be nice.
2. Don't be sneaky or envious.
3. Be generous
4. Retaliate appropriately

The 3-Prisoner's Dilemma has the Payoff Matrix of the following.

State (A, B, C)	**Payoff (A)**
(C, C, C)	6
(C, D, C) (C, C, D)	3
(C, D, D)	0
(D, C, C)	8
(D, D, C) (D, C, D)	5
(D, D, D)	2

Dominant Strategy

U(Defect) = (8+5+5+2)/4 = 5

U(Cooperate) = (6+3+3+0)/4=3

Nash Equilibrium

(D, D, D) = 6

However, this is not *Pareto Optimum*

Maximum Social Welfare

(C, C, C) = 18

It is beneficial to trigger (C, C, C) for long-term rewards. In the classic Prisoner's Dilemma, TitForTat is the most successful strategy, winning the Axelrod's Tournament twice in a row. This strategy starts with a cooperation and imitates the previous moves of the opponent. If the opponent stays cooperative, TitForTat will cooperate. For any defect, it will immediately reply with a defect. Because it makes use of the opponent's previous move, there are multiple ways to implement the idea of TitForTat in 3-person Prisoner's Dilemma. Hence, I came up with four strategies:

1. If two opponents had the same previous move, follow their move. Else, Defect. Start with Cooperate
2. If two opponents had the same previous move, follow their move. Else, Cooperate
3. If two opponents had the same previous move, follow their move. Else, Defect. Start with Defect
4. Get the probability of cooperation for each opponent based on the full history. If they are both are likely to defect, Defect. Else, cooperate. Start with Cooperate.

Performance of each strategy against the 6 given strategies were measured. The tournament defined in the given code was held 1000 times each to find frequency of rank.

Model	Rank 1	Rank 2	Rank 3	Rank4	Rank 5	Rank 6	Rank 7
1	52.8%	27.9%	17.3%	2.0%	0.0%	0.0%	0.0%
2	18.3%	20.8%	22.7%	19.4%	10.9%	6.2%	1.7%
3	0.0%	0.0%	0.5%	3.3%	13.7%	32.1	50.4%
4	47.5%	24.6%	17.9%	8.0%	1.9%	0.1%	0.0%

Strategy 1 and 4 performed well out of the 4 models. Both models were ranked 1st for approximately half the time. Hence, I decided it was worth exploring more with the two models.

Model 1

Idea: Start with Cooperate. If two opponents have the same previous move, imitate their previous move. Otherwise, Defect.

This agent will begin with a cooperation in attempt to trigger cooperation from other agents to reach maximum social welfare. However, if both opponents defect together, giving the agent a sucker's payoff, it will present defect the next round. If both opponents have cooperated the previous round, it will cooperate the next round, hoping to reach maximum social welfare. If the opponents do not have the same move, it will defect as it is the dominant strategy.

Model 2

Idea: Start with Cooperate. Get probability of cooperation for each opponent based on the full history. If they are both are likely to defect, Defect. Else, cooperate.

This agent begins with cooperating in attempt to trigger cooperation from other agents to reach maximum social welfare. For each round afterwards, the agent will calculate the probability of each opponent cooperating by examining the full history of the opponent's history. A random number generated between 0 and 1 is generated to predict that opponent's next move. If both opponents predicted next move is to defect, the agent will defect. Otherwise, it will cooperate in attempt to trigger cooperative behaviour from the opponents.

Detailed Experiment

To further explore the ability of these models, other classical strategies and agents from GitHub and were added, making it the total player number of 22.

```
int numPlayers = 22;
Player makePlayer(int which) {
    switch (which) {
    case 0: return new SoftT4TPlayer();
    case 1: return new HardT4TPlayer();
    case 2: return new Trigger();
    case 3: return new AlternatePlayer0();
    case 4: return new AlternatePlayer1();
    case 5: return new SoreLoser();
    case 6: return new Nasty2();
    case 7: return new Nice2();
    case 8: return new EncourageCoop1();
    case 9: return new EncourageCoop2();
    case 10: return new Mundhra_Shreyas_Sudhir_Player();
    case 11: return new SimpleMajorityPlayer();
    case 12: return new ExpectedUtilityPlayer();
    case 13: return new HybridPlayer();
    case 14: return new WILSON_TENG_Player();
    case 15: return new DongwooOnePlayer();
    case 16: return new NicePlayer();
    case 17: return new NastyPlayer();
    case 18: return new RandomPlayer();
    case 19: return new TolerantPlayer();
    case 20: return new FreakyPlayer();
    case 21: return new DongwooTwoPlayer();
```

Model	Top 1	Top 2	Top 3	Top 4	Top 5
1	0%	0%	0%	0%	0%
2	34.4%	26.2%	16.6%	9.9%	6.9%

Model 1 performed poorly with other models while model 2 performed well.

Model 2 was therefore decided as my final model. Below is a brief explanation of the implementation of Model 2.

Justification

The possible reason why Model 1 did not perform too well might be due to its strategy of defecting unless the two opponents have both cooperated in the previous rounds. Being a less generous agent, unlike fixed strategies that were compared with in the beginning, dynamic strategies that other students have made are likely to not tolerate such behaviour. These agents are built in attempt to be in the top scoreboard, so they are not likely to let one agent exploit themselves: hence, leading to frequent defective behaviour. Hence, Model 1 was not able to reach cooperation easily and being a version of Tit-for-tat, it is made to draw, not to win. Hence, it did not perform well.

Model 2 performed well, and this might be because it did not retaliate to Defect immediately. It only retaliates when the opponent has a history of defecting more often and is likely to defect the next round. Additionally, it only defects when both opponents are deemed to defect. Hence, it is more generous. It also aims to draw instead of trying to win over the opponents, making it nice. Hence, it overall has a larger chance of reaching cooperation state with dynamic agents who are most likely aiming to reach cooperation as well.

Final Model

The model satisfies the strategies (sort of) that Axelrod suggested mentioned above. It is nice. It starts off with cooperation and it will not betray by defecting first. It will not exploit you even if you show a sign of always cooperating. The model is generous because it will not punish you for defecting unless you have an history of defecting more than cooperating. The strategy is quite simple. If you show tendency to cooperate more, the model cooperates. If you show tendency to defect more, it will defect as well.

Model 2 has another function called getCoopRate.

```
int getCoopRate(int n, int[] history) {
    int i = 0;
    for(int action:history) {
        if(action==0) i++;
    }
    if(Math.random()<(i/n)) return 0;
    return 1;
}
```

This function basically takes the history of the opponent and calculates the rate of cooperation. The rate is considered as the probability of cooperating next time. If a randomly generated digit it less than the probability, the opponent is predicted to cooperate. Otherwise, it is predicted to defect. Then, this is used in the main selectAction function.

```
int selectAction(int n, int[] myHistory, int[] oppHistory1, int[] oppHistory2) {
    if(n==0) return 0;
    if(this.getCoopRate(n, oppHistory1)==1 && this.getCoopRate(n, oppHistory2)==1)
        return 1;
    return 0;
}
```

The selectAction function will return 0 when n is 0, meaning it will cooperate in the first round. From the second round, it predicts the opponent's move by calling getCoopRate and return 1 if both equals to 1. This means the agent will defect when both opponents are predicted to defect. Otherwise, it will cooperate.

To conclude, I believe that this agent is likely to be **not exploited** by other agents and is most likely to **maintain cooperative behaviour** to get a high score.