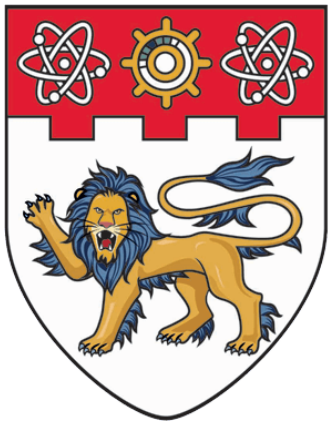

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4046 Intelligent Agents
Assignment 2

Name	Kong Jie Wei
Matriculation Number	U1720017C

Table of Contents

1. Description and Design of Agent

The design of my agent largely follows the suggested rules for succeeding in Robert Axelrod's tournament as shown on Module 8.1 – Multiagent Interactions of our lecture slides. The suggested rules are:

- Be nice – Start by cooperating, and reciprocate cooperation
- Retaliate appropriately – Always punish defection immediately, but don't overdo it
- Don't hold grudges – Always reciprocate cooperation immediately
- Don't be envious – Don't play as if it were zero sum

In this Three Player Repeated Prisoners' Dilemma, integer "0" is used to represent cooperation, and "1" to represent defection.

1.1 Start by cooperating

```
// start by cooperating in the first round
if (n==0) return 0;
```

Figure 1.1 Start by cooperating

Figure 1.1 shows my agent's strategy for the first round, which is to cooperate. The motivation behind this is to signal to other agents that my agent is keen on cooperation and to be nice.

1.2 Strategy for second round

```
if (n==1) {
   // check if any opponent defected in the first round
   if (oppHistory1[n-1] == 0 && oppHistory2[n-1] == 0) return 0;
   else return 1;
}
```

Figure 1.2 Check if any opponent defected in the first round

My agent's strategy for the second round is to look at what my opponents did in the first round, and if any of them defected, my agent will defect in the second round. This strategy is shown above in Figure 1.2. This aligns with one of the suggested rules for succeeding, which is to punish defection immediately.

1.3 Retaliate if opponent defected recently

```
// after first 2 rounds
// check if any opponent defected recently in the previous 2 rounds
if (oppHistory1[n-1] == 0 && oppHistory2[n-1] == 0
   && oppHistory1[n-2] == 0 && oppHistory2[n-2] == 0) {...}
// defect to save myself if any opponent has recently defected
else {
   return 1;
}
```

Figure 1.3 Retaliate if opponent defected recently

Figure 1.3 above shows my agent's strategy after the first 2 rounds. It first looks at the recent histories of my opponents, the previous 2 rounds. If there were any recent defections in the previous 2 rounds, my agent will defect. This aligns with one of the suggested rules for succeeding, which is to punish defection immediately.

1.4 Cooperate if opponent cooperated frequently

If there were no recent defections, my agent proceeds to look further back into my opponents' histories, to see their track record for cooperation.

```
// calculate opponents' defection rate
double opp1DefectRate, opp2DefectRate;
opp1DefectRate = opp2DefectRate = 0;
for (int i = 0; i < n; i ++) {
   opp1DefectRate += oppHistory1[i];
   opp2DefectRate += oppHistory2[i];
}
opp1DefectRate /= n;
opp2DefectRate /= n;
```

Figure 1.4.1 Looking at opponents' histories

By looking back at my opponents' histories, it enables my agent to calculate my opponents' rate of defection, as shown above in Figure 1.4.1.

```
double DEFECT_THRESH = 0.1; // max defection rate to tolerate
if (opp1DefectRate > DEFECT_THRESH || opp2DefectRate > DEFECT_THRESH) {...}

//cooperate if both opponents less than my defection threshold
return 0;
```

Figure 1.4.2 Tolerance for defection

My agent has a defined variable `DEFECT_THRESH`, which defines its tolerance to its opponents' rate of defection. My agent will cooperate if its opponents have a good track record for cooperation. This is to reciprocate cooperation and is shown above in Figure 1.4.2. The motivation behind such a low value for the rate of defection is to show that my agent has a low tolerance for defection and will retaliate appropriately.

1.5 If opponent defected frequently

If any of my opponents defected more than my defined threshold, my agent will proceed on to consider whether to forgive my opponents or not.

```
if (opp1DefectRate > DEFECT_THRESH || opp2DefectRate > DEFECT_THRESH) {
   double FORGIVE_THRESH = 0.1; // probability of forgiveness

   // consider forgiving
   if (Math.random() < FORGIVE_THRESH) return 0;

   // don't forgive
   else return 1;
}
```

Figure 1.5 Consideration for forgiveness

My agent has a defined variable `FORGIVE_THRESH`, which defines my agent's probability of forgiving my opponents should they defect more than the defection threshold. The motivation behind such a low probability is to show that even though my agent does forgive my opponents for defection, it does not mean that they can easily take advantage of my agent's kindness and my agent will retaliate appropriately.

2. Source Code of Implemented Agent

```
class Kong_JieWei_Player extends Player {
   int selectAction(int n, int[] myHistory, int[] oppHistory1, int[] oppHistory2)
{
      // start by cooperating in the first round
      if (n==0) return 0;

      if (n==1) {
         // check if any opponent defected in the first round
         if (oppHistory1[n-1] == 0 && oppHistory2[n-1] == 0) return 0;
         else return 1;
      }

      // after first 2 rounds
      // check if any opponent defected recently in the previous 2 rounds
      if (oppHistory1[n-1] == 0 && oppHistory2[n-1] == 0
         && oppHistory1[n-2] == 0 && oppHistory2[n-2] == 0) {
         // calculate opponents' defection rate
         double opp1DefectRate, opp2DefectRate;
         opp1DefectRate = opp2DefectRate = 0;
         for (int i = 0; i < n; i ++) {
            opp1DefectRate += oppHistory1[i];
            opp2DefectRate += oppHistory2[i];
         }
         opp1DefectRate /= n;
         opp2DefectRate /= n;

         double DEFECT_THRESH = 0.1; // max defection rate to tolerate
         if (opp1DefectRate > DEFECT_THRESH || opp2DefectRate > DEFECT_THRESH) {
            double FORGIVE_THRESH = 0.1; // probability of forgiveness

            // consider forgiving
            if (Math.random() < FORGIVE_THRESH) return 0;

            // don't forgive
            else return 1;
         }
         return 0;
      }
      // defect to save myself if any opponent has recently defected
      else {
         return 1;
      }
   }
}
```

Figure 2 Implementation of my agent

Figure 2 above shows the full source code and implementation of my agent.

3. Evaluating Agent's Performance

3.1 Against Nice Player

A Nice Player's strategy is to always cooperate. When my agent was against a Nice Player **only**, both of them starts off with cooperation. Subsequently, my agent looks into its opponents' histories, which in this case is filled with cooperation, and cooperates as well. This results in a tie between my agent and a Nice Player. The tournament results are shown below.

```
Tournament Results
NicePlayer: 36.0 points.
Kong_JieWei_Player: 36.0 points.
```

3.2 Against Nasty Player

A Nasty Player's strategy is to always defect. When my agent was against a Nasty Player **only**, my agent cooperates at the start while the Nasty Player defects, resulting in my agent losing out. Subsequently, my agent looks into its opponents' histories, which is filled with defection in this case, and defects as well. Additionally, my agent may occasionally decide to forgive the Nasty Player, resulting in further losses incurred by my agent. Despite my agent losing to Nasty Players, it is able to defend itself relatively well (points scored that round is close to the Nasty Players). Afterwards, my agent is able to earn much more points from cooperation when it plays with another agent like itself. Due to this, my agent will always do better than Nasty Player, and an example of such tournament is shown below.

```
NastyPlayer scored 2.0 points, NastyPlayer scored 2.0 points, and NastyPlayer scored 2.0 points.
NastyPlayer scored 2.030303 points, NastyPlayer scored 2.030303 points, and Kong_JieWei_Player scored 1.979798 points.
NastyPlayer scored 2.0631578 points, Kong_JieWei_Player scored 2.0105264 points, and Kong_JieWei_Player scored 2.0105264 points.
Kong_JieWei_Player scored 6.0 points, Kong_JieWei_Player scored 6.0 points, and Kong_JieWei_Player scored 6.0 points.

Tournament Results
Kong_JieWei_Player: 24.00085 points.
NastyPlayer: 12.123764 points.
```

3.3 Against Random Player

When my agent is up against a Random Player **only**, one who defects 50% of the time, it is highly likely that the Random Player agent's defection rate will surpass my agent's tolerance for rate of defection, and my agent will retaliate appropriately. Due to this, my agent will always do better than Random Player, and an example of such tournament is shown below.

```
RandomPlayer scored 4.0970874 points, RandomPlayer scored 3.9029126 points, and RandomPlayer scored 4.0970874 points.
RandomPlayer scored 2.6095238 points, RandomPlayer scored 2.5619047 points, and Kong_JieWei_Player scored 5.3238096 points.
RandomPlayer scored 1.0285715 points, Kong_JieWei_Player scored 3.552381 points, and Kong_JieWei_Player scored 3.552381 points.
Kong_JieWei_Player scored 6.0 points, Kong_JieWei_Player scored 6.0 points, and Kong_JieWei_Player scored 6.0 points.

Tournament Results
Kong_JieWei_Player: 30.428572 points.
RandomPlayer: 18.297089 points.
```

3.4 Against Tolerant Player

A Tolerant Player's strategy is that it looks at his opponents' histories and defects if at least half of the other players' actions have been defects. When my agent goes up against a Tolerant Player **only**, both of them always cooperates which results in a tie at the end of the tournament. The result of the tournament is shown below.

```
Tournament Results
TolerantPlayer: 36.0 points.
Kong_JieWei_Player: 36.0 points.
```

3.5 Against Freaky Player

A Freaky Player's strategy is that it decides to be either a Nice or Nasty Player at the start, and my agent will react accordingly as stated above.

3.6 Against T4T Player

A T4T (Tit-for-tat) Player's strategy is that it cooperates at the start. Subsequently, it randomly picks one of its opponents and return their previous action. When my agent is up against a T4T Player **only**, since both my agent and it cooperates at the start, this cooperation carries on till the end, resulting in a tie between my agent and a T4T Player. The result of the tournament is shown below.

```
Tournament Results
T4TPlayer: 36.0 points.
Kong_JieWei_Player: 36.0 points.
```

3.7 Three Player Repeated Prisoners' Dilemma Results

As there are some randomness involve in some of the agents' strategies, each tournament has to be repeated numerous times to effectively gauge my agent's performance against them.

```
Average of 5000 rounds ranking
Kong_JieWei_Player: 169.33282 points.
TolerantPlayer: 164.74803 points.
T4TPlayer: 163.69756 points.
NicePlayer: 151.65402 points.
FreakyPlayer: 146.3334 points.
NastyPlayer: 140.47935 points.
RandomPlayer: 133.9694 points.
```

Figure 3 Average results of 5000 tournaments

Each tournament can consist of between 90 to 110 rounds, and the above Figure 3 shows the result of running 5000 of such tournaments. Figure 3 also shows the ranking of each agents, as well as their average points across the 5000 tournament runs. It can be observed that my agent performs very well when competing against the other players as mentioned above. This is due to it incorporating the rules for succeeding in Robert Axelrod's tournament as listed above, resulting in my agent being able to cooperate with agents who reciprocate cooperation and defend itself from agents that defects often.