1. (3 points, mandatory) propose three possible nontrivial reasonable ways how to define the state in the game.

```
1> 'Ace Policy'
 if usable Ace (Player hand contains Ace):
     (All Player sums <= 8 ➔ 8) & (All Player sums >= 21 ➔ 21) & (All Dealer sums >= 18 ➔ 18):
         Player sum 8 * Dealer sum 1-18 * (2 actions)
         Player sum 21 * Dealer sum 1-18 * (2 actions)
     Else:
         Player sum 9- 20 * Dealer sum 1-18 * (2 actions)
  if No usable Ace (Player hand doesn't contain Ace):
     (All Player sums >= 8 ➔ 8) & (All Player Sums >= 21 ➔ 21) & (All Dealer sums >= 18 ➔ 18):
         Player sum 8 *Dealer sum 1-18 * (2 actions)
         Player sum 21 * Dealer sum 1-18 * (2 actions)
     Else:
         Player sums 9- 20 * Dealer sum 1-18 * (2 actions)

2> 'All Sums Policy'
All Player sums * All Dealer sums * (2 actions)

3> 'Dealer Policy'
(All Player sums <= 6 ➔ 6) & (All Player sums >= 17 ➔ 17) & (All Dealer sums <= 5 ➔ 5) & (All Dealer sums >= 21 ➔ 21):
     Player sum 6 – 17 * Dealer sum 5- 21 * (2 actions)
```

---

2. (1.5 points) For each state space representation from 1, provide a rough estimate of the overall number of states. You cannot just guess a number; you have to justify it somehow (e.g., by calculation). You do not need to provide an exact number; however, your estimate should not be too far from the true count. If you are not sure how to calculate the number of states, you can write a program that counts them for you and submit the code together with your report.

note: Code used to calculate is commented out in sarsaagent.py line(32-56)

1> *'Ace Policy'* : 540

2> *'All Sums Policy' :* 760

3> *'Dealer Policy' :* 352

---

3. (2.5 points, mandatory) Pick one of the state space representations you proposed in 1. Explain why you consider it the best one and answer the following questions. Does this representation capture all information that can be used for agent decision? Or is there any simplification? If yes, will the simplification influence the result (final policy, utility values)? If yes, how much will the result be influenced? Can you use exact methods (value iteration/policy iteration) to solve the game? If yes, how? If not, why? In one or two sentences explain your choice of discount factor and the number of games that you need to learn the Q-values.

I ended up with the 'All Sums Policy' because any other generalization I used left too much room for error or discrepancy. I had assumed that there was a statistical 'key' to the game. i.e. The ace being in play, or the dealer policy of hitting until a sum of 17. But in the end having the exact observation recorded map directly to the Q states produces the best and most consistent result for me, which allows for the agent to have enough information and exploration space to learn and choose the optimal policy

Implementing policy iteration through the SARSA agent was successful but only when I did not simplify. Simplification caused my algorithm to be inconsistent with its decision making. This is caused by the algorithm grouping together states that cause it to overgeneralize and produce wildly different final policies for every run.

I chose a discount factor of .97 and 80000 epochs. This is to prioritize rewards in the distant future so as to maximize the final accuracy and precision after a learning and exploratory period. I sacrificed initial rewards for greater overall efficacy in the end.

____

6. Test your code and provide an experimental evaluation. Compare the random strategy (provided), the dealer strategy (provided), the result from 4 and the strategy learned by SARSA.

My SARSA agent has a consistent average utility of ~.04 - ~.08 for the last 8% of evaluations. Which is comparable to that of the dealer agent. And better than the random agent. Although it takes much longer (more epochs) to gain this good average utility. As mentioned before, I did initially expect that generalizing the states by usable Ace or not, would be the key to quickly learning. But it turned out that the ace was not a good benchmark to base the learning on. It is either not statistically significant to the outcome of the game, or, too variant of a quality to base the state space on.

For the example case of club nine, diamond jack and spades two in your hand and dealer has club four my SARSA agent produces a best Q value of 0.903366 for stay. While the diamond ace and spades five and dealer spades ace has a best Q value for -0.43695 for stay. The agent treats the ace as its 11 state unless there is a sum over 21 then it revert to 1. Even with this quality these two states have effectively converged to the same decisions and the agent has proven effective.