Implementing Basic Poker Strategy (Week 1):

To create the skeleton structure of the bot, we first implemented strategies that are common in standard Texas hold'em that we could adjust for auction hold'em. These strategies included determining which hands to fold or raise with preflop as well as deciding action post-flop. In the first week, we decided to split up our work so each team member would be working on a specific part of the bot, and then we would synthesize our code into one final submission for the week. Chase would determine action post-flop, Cole would analyze opponent betting patterns and determine a range based on bet size, Lawrence would determine auction betting size, and Owen would determine action pre-flop.

We determined that the best course of action to determine preflop action was to proceed as close to GTO as possible. After conducting research about which hands to play preflop and optimal betting strategies, we brainstormed efficient ways to store all possible two-card hands. Since hands with the same ranks but different suits have the same strength preflop, we can distinguish these hands as "offsuit" and hands with the same suits as "suited". As we are using a dictionary to store these values, this cuts the number of keys down from 52*51 = 2652 to 13*13 = 169. The values associated with these hands are numbered from 1 to 169, the lower numbers indicating stronger hands, with "AAo" having a value of 1. We then wrote a function that maps our given hand in ["rank1, suit1","rank2, suit2"] format to "AAo". We then either raised or folded depending on the strength of our hand because just calling would show weakness. We then decided to make our ranges for cases if preflop action were to continue past a big blind call.

For this, we implemented a mathematical function that ended up being of the order of the cube root of the opponent's bet size, which shrunk our calling range the more the opponent bet.

For our auction action, we implemented a Monte Carlo simulation that determines the probability of our team winning with and without an auction, then determining a bid size based on these probabilities. This would later be tweaked to account for a straight or flush draw with semi-bluffs. Our post-flop action is similar, also using a Monte Carlo simulation to determine our hand strength if we win or lose the auction. This is done by giving the opponent two or three random cards and adding random cards to the board to make it full for every trial. We then compare our hand strength, generated using the eval7 library, to our opponent's to determine a winner for that trial. Throughout 125 trials, we then generate a percentage of how many times we win the hand. We then determine pot equity by dividing the amount we need to call by the total pot minus the amount we need to call. Using pot equity, hand strength, and a random number generator, we then determine when to raise, fold, check, and call.

Bluffing Strategies (Week 2):

We noticed that there were a lot of bots that had very tight ranges both pre-flop and post-flop, so we decided to explore bluffing strategies. We also noticed that teams almost always never checked when they had good hands. To exploit this, we added what we called "1 check bluffs" and "2 check bluffs." This means each time the opponent checked, we were more likely to bluff. When the opponent checked 1 time, we bluffed 30% of the time (with a slight decrease if we were the big blind instead of the dealer). When the opponent checked twice, we bluffed 75% of the time as the big blind and 100% of the time as the dealer. This was very successful, especially the two check bluffs. We were able to exploit almost all teams that didn't use ML to

adjust throughout the game. For the teams that started to catch on to the fact we were bluffing, we kept track of our plus-minus resulting from bluffs. Once our bluff plus-minus fell below a certain threshold, we would then bluff ¼ of the time we were initially bluffing. This also protected us against teams that would rarely fold no matter their hand strength.

The other major change we made this week was to dynamically decrease trials in our Monte Carlo simulations if we were running low on time. We dynamically decreased from 125 to 100 to 50 trials as we ran low on time. We knew decreasing the amount of trials would lower the accuracy of our sims. To account for this, we added a "nit" factor that would decrease our hand strength accordingly to account for this lack of precision and prevent very loose calls (decreased by 3% and 6% from 100 and 50 trials respectively). Finally, to ensure we were only continuing in hands with very big pots when we had strong hands, we subtracted an additional 3% from our hand strength when our contribution was over 100. These changes were not big changes, and so we all went on a Zoom call and did the changes together.

Analyzing Opponent Betting Strategies (Week 3):

In week 3 we realized that opponents were bluffing just like we were, so we implemented bluff catchers to "catch" and exploit the bluffs. We checked how many times the opponent bet straight after winning the auction and also how many times we checked and then they immediately bet. These are when we found the opponent bluffed the most, as they might think we see them as extra strong with a bet after an auction or that we seem extra weak by checking. Thus, every time we thought they were bluffing, we increased an "unnit" factor by a small amount so that we would be calling more (lowing pot equity needed to call). We subtracted this "unnit" factor from the pot equity (essentially decreasing the hand strength needed to call).

We also had an “aggressive” variable to see if an opponent was aggressive. If an opponent was deemed aggressive (they bet pot a lot), we would increase the “unnit” factor even more so we would be calling more. For each unnit, we increased more from the pot equity needed to call if we had 2 cards vs 3 because we deemed the opponent would be more aggressive if they had 3 cards (winning the auction). We kept track of how many calls we made with the increased unnit factor and kept track of the plus-minus from it, and if we deemed to lose too much based on the unnit factor, we just divided the unnit factor by 2. All the unnit factors stacked on top of each other, but we made sure not to unnit by more than 0.25, so we never increased unnit both because of the opponent auction bluff and the opponent check bluff. These changes were not big changes, and so we all went on a Zoom call and did the changes together.

Final Touches (Week 4):

For the last couple of days leading up to the competition, our team made some final touches to our bot. These included introducing the semi-bluff, which was initially an idea to gauge the aggressiveness of the opposing bot. We defined a semi-bluff as betting larger than we would normally while being on a flush draw or straight draw. In eval7, our draw hitting percentage while being on flush or straight draw is usually greater than 0.250, so we set our semi-bluff threshold to be greater than 0.250 (25% straight/flush draw hit %) when a flush or straight draw is present, and bluffed with probability accordingly. Seeing as the semi-bluff proved to be effective against most opponents, we decided to keep this as an action throughout the entire game instead of the first one hundred hands. We also changed our range preflop to raise with slightly more hands to counter loose opponents, and we reordered some entries in the dictionary to increase the value of suited, connected, and suited connected cards. With this, we

felt that our bot was performing as well as it could, and our final version was submitted. These changes were not big changes, and so we all went on a Zoom call and did the changes together.