Project 3 Summary:

For this project, we started off by writing a few lines of code that formed how the game operates. This means we used randomness to flip a coin. The random function returns either a 0 or 1. A 0 means the coin flip returns “tails”, and a 1 signifies “heads”. Next, we set up a points system. At the beginning of the coin flip function, we initialize the points to zero as well as the turns. While turns is less than 144, the algorithm flips coins and awards/removes the points based on the coin that was flipped, and the side it landed on. At the end of the 144 turns, the function returns how many points were awarded in total.

At first, we used randomness to make the coin choice either a dime or quarter 50% of the time. We observed how many wins/losses we gained from a completely random algorithm before moving on. With this method, we would win about 50% of the time. We needed to figure out a strategy to raise this proportion in order to win as much as we can.To do this, we started by setting the algorithm to flip a quarter ONLY if the points were less than zero. Otherwise, we would flip a dime. Through this idea, we would only risk our two points when we needed to reach a total point value of 1. If we were above zero, we would flip dimes to try to secure our position above 1, without risking a negative dip of two points in one turn. It worked pretty well, making our wins about 60% of the time. This seemed to be a good working strategy. We tried to change when exactly we should flip a quarter in order to make it even better through parameters based on which turn it was, as well as how many points we had.

In order to do this, we needed to take a look at how the games were being played. We set up our main to display the turn number as well as the number of points we held. We ran 100

trials of the game, and observed where some of the flaws were in our code. In some cases, the values would go super negative as a result of flipping quarters endlessly as long as the points were negative. The values would dip as low as -40 points or more, sometimes just before the end of the game, effectively throwing the game in a lot of cases.

By working backwards, we knew that we had to implement strategies to keep this from happening, and to keep the values as close to 1 as possible throughout the game.
We edited the code so if the values went below ~-25 in the middle of the game, we would flip only dimes to keep the values from dipping anymore.As it gets closer to the end of the game, we raise this value up until it is -18 at turn 125. Turn 125 is close to the end of the game, and it would be too risky to flip quarters and have the points dip too much here. By turn 137, if the points are still too low, like below -10, we need to make risks to flip quarters to get above 1 by turn 144. We have 8 turns left, so it would be possible to surpass 1 if we get enough good quarter flips and the point value is around -10.

By the time it is turn 137, if the point value is around -10 like we planned, it is crucial that we risk it all in order to get the value above 1. This is the reason we flip only quarters if the point value is less than 1 at turn 137. Each time the function runs, we add +1 to the “turns” value, which is initialized at zero. When 144 passes through the function happens, the function ceases the coin flip and returns our final value of points. Through our safeguarding strategy, we acquire about 65 wins per 100 games, reaching up into the 70s at some times.