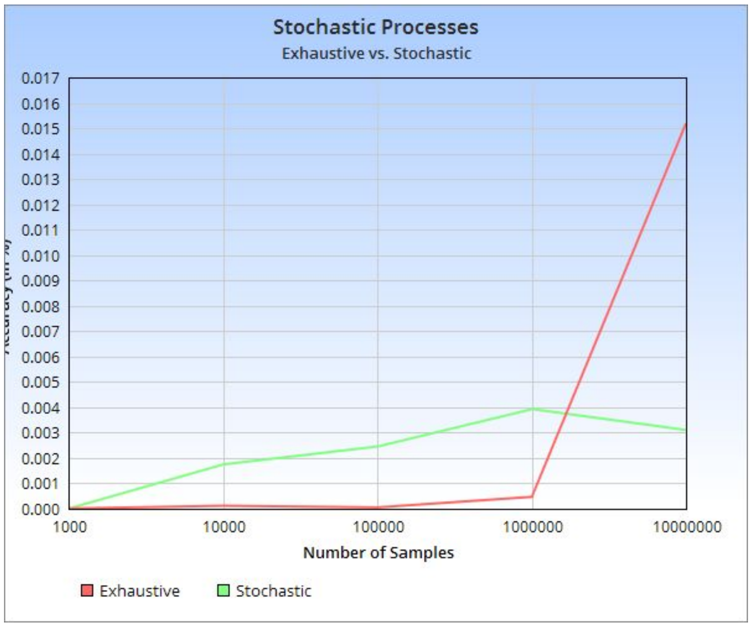
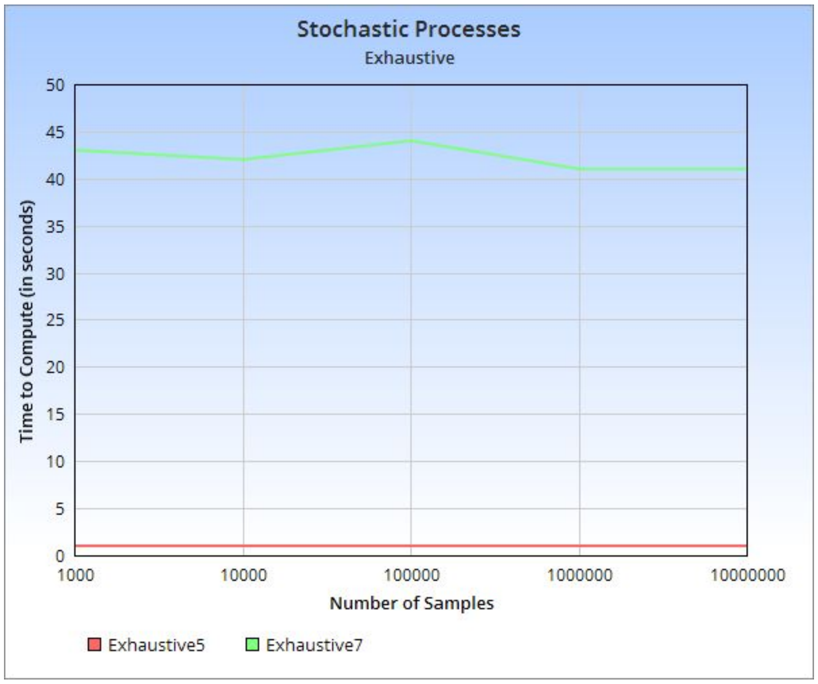
Ashton Schmidt
Cs 2420 Assignment 8 Analysis

1. What did you learn about stochastic processes and random number generation?
    a. I Learned That the way that you generate random numbers is key when you are wanting to get a random number. Because if you use anything that has a repeating sequence such as time your “random” numbers will not be random. In real life there is not always going to be numbers that are in order or things that are in order. Because of that fact it is necessary to test real world applications for the software that we are developing rather than just the best case scenario because that will not always happen and there is a small percentage of times were the best case scenario is achieved. Computer Scientists use random numbers and develop ways to get completely random numbers to get a more accurate result for real world problems, so that we can better analyze the world and the scenarios that we encounter on a daily basis.
    b. As to the Stochastic process I learned more about the real world percentages of wins. This is important because when you are playing poker the order of the deck and the hand you're dealt is unpredictable in real life, so why would we want percentages for something that doesn't simulate a real world problem? This is why the Stochastic method is not only applicable but rather necessary when analyzing code and the percentages that are done. Although knowing what your chances are with all cards of the deck can be nice and somewhat promising, knowing the chance of getting that hand based on randomness is better because it is more along the lines of something that you would actually encounter at a table in Las Vegas.

2. 

This graph shows us the difference between the Exhaustive and the Stochastic, as the number of samples increases we want more stability which the stochastic method provides, because it is not taking random numbers but rather checking every value so as the samples increase the accuracy should increase as well because there is a greater chance that the hands will be matches rather than just having random hands.

If you choose accuracy over time it would be slow but You would want it because you would get more accurate percentages of hands, rather than just having a fast algorithm. In our case when we are testing real world applications and want to know the percentages of winnings and how they differ with different number of hands. This implementation does not require the timing to be extremely fast, so we would pick having an accurate percentage over speed optimization for this program.



This graph shows the difference of time it takes to compute different numbers of samples, when we have 7 cards, it needs 7 nested for loops in order to get every card. Which is way more than 5 nested for loops and will obviously take more time to compute. Being that our time is where it is which is about 42 sec (7*6) I calculate a nested loop of 9 would be 72 seconds(9*8). Which would make sense because we are nesting 2 more loops to the same amount of values.

3. Software Development Log
    a. We spent probably just a little over 15 hours on this programming assignment.
    b. The most time consuming part of the code I believe was checking for all of the values because you had to think of all of the edge cases and then break it down to return the highest value. (example if you had two hands with each two of the same pairs you would have to get the high card of the two hands and compare those.)
    c. Something that I believe I can do better in the future is probably create more reusable code and use more helper methods, because due to our implementation I ended up writing a couple bits of the same code where I could have cut down and made more helper methods just to make my code a little more clear and human readable.
    d. We allocated enough time to finish the project and have had some extra time to spend working on the project which has been extremely nice, because that allows us to tweak our code and make sure that things are running correctly rather than just trying to turn in something that runs.
4. Texas Hold'em Analysis
    a. Based on our analysis of Texas hold em, we realized that it is just like poker you just have to add the two cards that you were given to the tables 5 to create the best hand possible. This is also done with the opposing player with the same 5 cards on the table but 2 different cards than the opposing player and the table. When these hands are added you can compare the two hands as if it were just a poker game of 7 card draw.
    b. The top 10 best hands based on winning percentage are Ace Ace, King King, Q Q, A K, JJ, AQ, KQ, AJ, KJ, A 10. When you get these cards you can start to get a little happy in your head because you have a good percentage of winning, but as it is gambling there is a chance that the house or opposing player had just a card better than you.

5. Thought Problems
    a. The random number generator is extremely important when using it with the stochastic experiment, because if the random number generator has any sort of cycle then the numbers will no longer be random and then you

encounter a problem that is not a real world problem because in real life things are random and the chances are extremely extremely small that there will be a cycle of how something is done when it should be random. How important is the random number generator to a stochastic experiment?

b. Our poor random number generator did as expected and didn't do well at all in comparison to Java's, because it was meant to be poor so we had it return 1 which is a random number but it is repeating over and over and will forever thus no longer making it a random number. In comparison to Java's random number generator our implementation did quite will it took us 114758 times to generate all possible values from 0-10000 which is good because it is higher thus actually getting random numbers, because if it took the same it would mean that numbers aren't repeating making the numbers not random. We also had a just about even split of odd_even pairs: 5000 and even_odd pairs: 4999 which is what is desired. According to lab it is extremely hard to get the other two possibilities to split somewhat evenly so this is the closest that we got to Java's implementation.

c. Other tests that should be in the check random tests could possibly be, comparing two numbers in different parts of the numbers we want to test and then count how many times that number was generated and compare the amounts, by creating a loop that iterates over the generator a lot of times and see if there is a trend or not, and if there is not a trend then we can see that there actually are random numbers being generated.

d. When comparing our random number in comparison to Java's random number generator, Java created more of a variety of random numbers that were able to be used, whereas because we had a slit of even odd and odd even , and no odd odd or even even we got less of a variety of numbers. Our random number generator did its job of creating random numbers it could be better to create more of a variety of numbers thus making it more applicable to real life scenarios.

Texas Holdem |||| 2 hand	Probability
(Ace Spades, Ace Hearts)	85.24%
(King Spades, King Hearts)	67.60%
(Queen Spades, Queen Hearts)	80.14%
(Ace Spades, King Hearts)	65.94%
(Jack Spades, Jack Hearts)	77.59%
(Ace, Queen)	65.28%
(king, Queen)	63.32%
(Ace, Jack)	65.54%
(king, Jack)	63.95%
(Ace, 10)	66.24%