John Anderson 009200893

CS 149 Section 3

HW #5

Part 1)

1. 1 thread is never failing, so It takes me more than one threads to result in failure (2+ threads will give non zero number). Increasing the number of threads and iterations will tend to make the fail count higher value higher. I think that the higher number of threads and iterations increases the probability that there will be collision with the critical section. Because the higher amount of total operations that is needed to be done in parallel will increases the chances of collision on the critical section resulting in an error.

2.The answer is related to the previous question in that if you reduce the amount of iterations then the total operations will decrease, resulting in a probability of collision to be smaller. Making the number for error be smaller form less collision and less operations.

Part 2)

3. formula (cost per operation = (runtime + (overhead?)) / number of operations). As the number of operations is increased then the average cost per operation is decreased a little. But is not always the cause because it depends on how much collision there is.

4. We can get an idea when the number of operations is large enough. From the formula in the previous problem.

5. the yield causes the current thread to relinquish the CPU. So, yield has a much slower time from the yield operation added on top of the normal operation time.

6. Using yield will take the extra time and will not give valid times unless you also know the time that they yield function uses and remove that form the total time. So, using yield in our situation will not give us valid times because we are not taking into account removing the time that the yield operation is adding to the total time.

Part 3)

7. The more iterations there are the shorter the average time per operations becomes. I think It follows this tread because the operations are time is taking more into account and less of the overhead and outside operations time or by the type of synchronization.

8.The atomic number one is working best for me. Maybe because it has a basic simple concept with little overhead and doesn't have much to any wasted time. Also maybe because it works in a lower level and doesn't use as much as the others inside of java itself.