**Report:**

**Test case 1 No I/O interaction long processes >10 ms:**

This case naturally produced a low throughput of 0.086 because each process was large and as a result took a long time to finally reach the exit state. Also because the processes had a long CPU time the average wait time is naturally large (29.42ms). This also shows with the 44.4 ms turnaround time as well. However because there was no I/O all processes finished in order that they arrived in the ready queue.

**Test case 2 No I/O interaction short Processes<3ms:**

Because there was no I/O interaction the processes completed their CPU execution in order in which they arrived at the ready queue. Naturally this test has a much higher throughput than the previous at 0.466667 and a significantly lower wait time at 2.6 ms.

**Test case 3: Same arrival Time mixture of long and short processes no I/O.**

This test was used to see if there were any natural biases to certain processes with FCFS. Because there was a bit of chaos when determining who arrived first when they all arrived at the same time overall longer processes got favored as they take a long time to execute thus increasing the wait time for shorter processes. This is shown by the calculated wait time of 23.142857 ms. This test also showed a smaller throughput with 0.17. and a longer turnaround time of 31 ms.

**Test case 4: Long Processes with high I/O frequency and long I/O duration:**

This test naturally produced the slowest turnaround time showing at about 67 ms with a throughput of 0.07. This is expected as these already long processes are constantly being interrupted and taken out of the CPU execution to perform long I/O tasks. The result was a painfully slow system. Now with I/O taken into account the order which they arrive in the ready queue has less weight on whether they finish earlier or later as the processes are interrupted mid execution.

**Test case 5: Long processes with High I/O frequency and short I/O duration:**

In this test I took the same input file as the previous test but heavily reduce the I/O duration but kept everything the same. As usual the exiting order is affected by the I/O bursts. However we figured reducing the I/O duration would help speed up the number of processes being executed however the results although better than the last test did not improve the throughput by much. The throughput was 0.10, the average turnaround time was 51.7 ms and the average wait time is 39.4 ms. This showing that although processes are not being terminated any faster the wait time for each process was much faster than the previous.

**Test Case 6: Short processes with high I/O frequency and low I/O duration.**

This test shows how I/O frequency affects turnaround time and throughput. With a higher turnaround time and throughput than its similar no I/O frequency tests. However I did notice that the waiting time for the test with high I/O frequency was less. This is because the processes are spending less time in the ready queue and more time in I/O. This was interesting as it was expected that the more I/O frequency has more waiting time.

**Test Case 7: I/O frequency = 1ms and I/O duration = 1 ms like in part 1d**

This test gave an average throughput of 0.30 , a wait time of 8.57 ms and a average turn around time of 17 ms. When looking at the log the once all processes were in the system they manage to stay inside for a long time. This is due to the high I/O frequency. However the average wait time appears to be better than some of other similar tests because the I/O duration was fairly brief.

**Test Case 8: Varying arrival time:**

By giving each process a very unique arrival with enough space between processes being ready the result is one of the most productive and efficient schedules. Although the through put slows down a but from the CPU waiting for ready processes. The wait time is very low because the ready queue is less congested. And the average turn around time is faster because the processes spend less time in the ready queue and complete as they do not have to wait for other processes of varying CPU lengths.

**Test Case 9: Short processes arrive first:**

In this test case I made sure that shortest jobs arrive first while longer jobs arrive last. Each job arrives one second after the previous ones. What I found as a slow average through put due to the shorter jobs being completed first while the longer jobs take longer. However the wait time would shorter than having the longest jobs first because the shorter ones are done and out of hte way early.

**Test Case 10: Long processes arrive first:**

Having long processes go first seems to congest the flow of processes in the system. This is shown by the low throughput of 0.137255. This is because it takes a long time to finish the larger processes. Because of this long time to finish the average waiting time is also high 26.714286 ms. The result was a set of processes that bottlenecked the entry to the running state.