Question 1

Version 1 Single Thread	Version 2 Double Thread	Version 2 Double Thread (2 CPU)	Version 3 3 Thread (3CPU)
184.6 secs	345.58 secs	105.75 secs	197.39secs
182.36 secs	347.00 secs	109.49 secs	217.48secs
182.25 secs	345.33 secs	122.61 secs	225.80secs
184.32 secs	342.22 secs	101.27 secs	211.92secs
183.22 secs	346.73 secs	98.41 secs	210.11secs
183.35 secs (mean)	345.37secs	107.51 secs	212.54 secs
183.2 Secs (median)	345.55 secs	105.75 secs	211.92 secs

Some of the fastest times are seen within the double threaded version of the code and I think that it is primarily due to the processor using 2 cpu. The 3 cpu version is moving slower than it but faster than the single processor version. I think that the logic within the code plays a huge role in that aspect. The times are very spaced out because the processes dont share information and they must wait until all threads complete the tasks before being completely finished running.

Question 2

Version 1 Single Thread	Version 2 Double Thread	Version 2 Double Thread (2 CPU)	Version 3 3 Thread (3CPU)
67.53 secs	21.88 secs	27.47 secs	19.79secs
72.11 secs	21.51 secs	28.81 secs	19.30secs
67.20 secs	21.00 secs	13.32 secs	19.50secs
66.79 secs	21.70 secs	20.56 secs	19.43secs
67.57 secs	21.75 secs	18.63 secs	19.64secs
67.04 secs (mean)	21.57secs	21.76 secs	19.53 secs
67.57 Secs (median)	21.88 secs	20.56 secs	19.50 secs

Within this matrix the fastest times were recorded within the 3 threaded version of the program.The slowest times were in the single threaded version of the program. The single thread version had 1 CPU and the 3 thread had 3 so that plays a major role. The 3 threaded version also split up more of the workload between each process which aided in the process working faster and computing CPU tasks quicker. The double thread version are closer in time proximity because they all use very similar logic and conduct tasks primarily in linear time.

The two matrices when compared to one another vary in time drastically. The first matrix with all single threads running concurrently was much slower since the workload wasn't split between different threads.