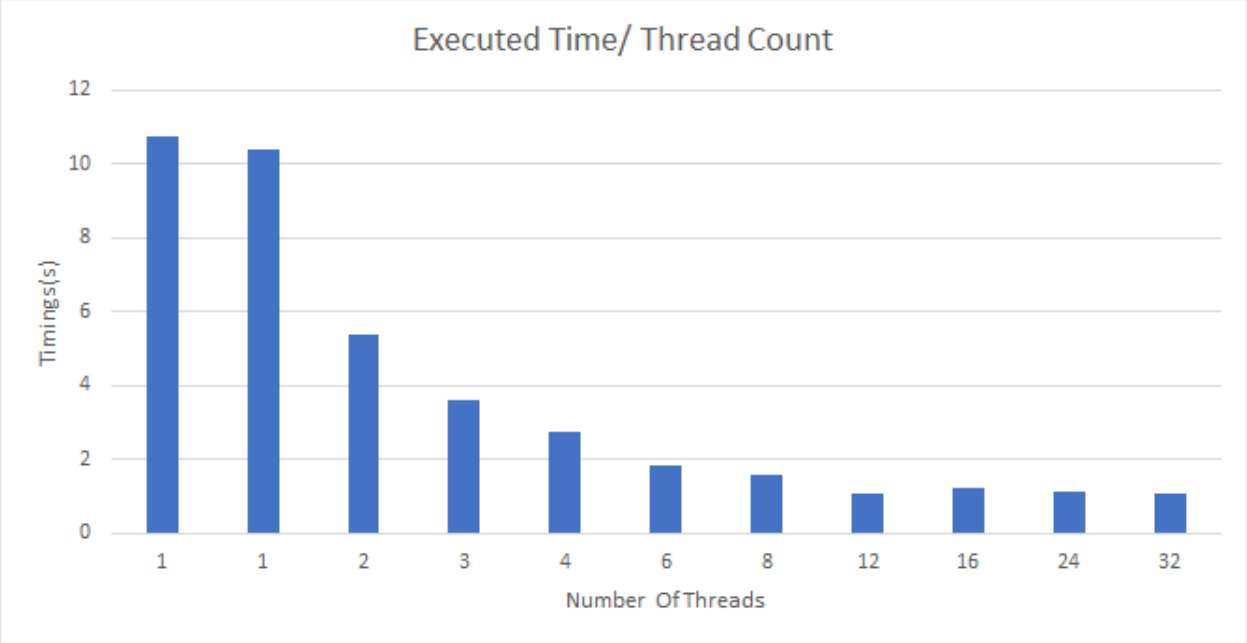
Question 2

A)

Threads	Timing(s)
1	10.737
1	10.378
2	5.375
3	3.601
4	2.736
6	1.824
8	1.591
12	1.093
16	1.248
24	1.136
32	1.104



B) I did not see N-times speed up for all values of N. Initially from threads 2 to thread 6 the program speeded up N-times compared to the original thread. Afterwards, threads 8 to 16 started to flatten out around the same value with no N-times speed up

C) The reason is if there are more threads than cores, then all cores are fully utilized so extra threads cannot run simultaneously. As a result threads run sequential instead of parallel

Question 4

Comments on results - I noticed that the ratio was not exactly perfect, this is due to a variety of reasons such as further optimization could have been made to improve speed. However, from thread 1 to 4, the ratio between observed and expected were similar but for thread 8 and 16 there was a huge disparity between the ratio. This can be explained by the limited number of cores the computer has and if the number of threads is larger than the cores then speed optimization starts to lose its effect since all cores are fully being utilized and there are still extra threads waiting to be run.

medium.txt			
Threads	**Observed Timing**	**Observed Speed Up Compared To Original**	**Expected Speedup**
Original Program	17.438	1	1
1	18.5396	0.940581242	1
2	9.5372	1.828419243	2
3	6.4623	2.698420067	3
4	4.8404	3.602594827	4
8	3.3837	5.153530159	8
16	2.7066	6.442769526	16

hard.txt			
Threads	**Observed Timing**	**Observed Speed Up Compared To Original**	**Expected Speedup**
Original Program	5.9392	1	1
1	6.2763	0.946290012	1
2	3.2539	1.825255847	2
3	2.1947	2.706155739	3
4	1.6686	3.559391106	4
8	1.122	5.293404635	8
16	0.8708	6.820395039	16

hard2.txt			
Threads	**Observed Timing**	**Observed Speed Up Compared To Original**	**Expected Speedup**
Original Program	5.993	1	1
1	6.2835	0.953767805	1
2	3.2584	1.839246256	2
3	2.1982	2.726321536	3
4	1.6457	3.641611472	4
8	0.826	7.255447942	8
16	0.7144	8.388857783	16