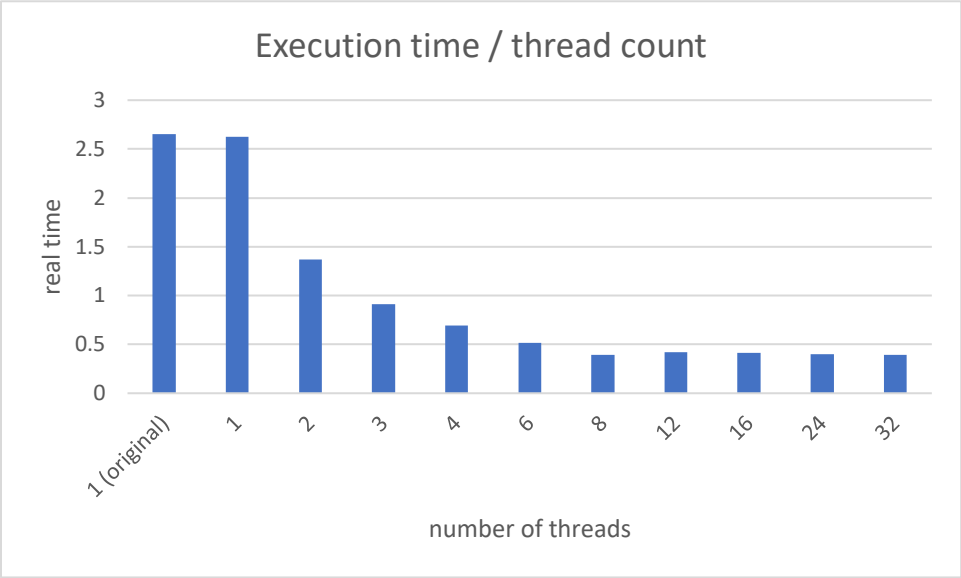
Question 1

Refer to *calcpi.cpp*

Question 2

a) All timings were done on gfx02-1 (i7-7700K CPU @ 4.20GHz)

Threads	Timing (s)
1 (original)	2.654
1	2.622
2	1.366
3	0.914
4	0.692
6	0.517
8	0.391
12	0.418
16	0.413
24	0.400
32	0.395



b) We see about an N times speed up (where N is the number of threads being utilized) up to and including the use of 8 threads, after which we see the speed increase plateau.

c) The 7700K which the timings were done on is a 4 core & 8 thread CPU, so we are actually slowing it down by making the CPU context switch more often between threads (in addition to the cost of creating & destroying more threads vs the cost of the work itself) once the plateau occurs (8 threads), which would explain why we didn't see an N times speed increase after 8 threads were assigned.

Question 3

Refer to *detectPrimes.cpp*

Question 4

a) All timings were done on gfx02-1 (i7-7700K CPU @ 4.20GHz)

medium.txt			
# threads	**Observed timing**	**Observed speedup compared to original**	**Expected speedup**
original program	17.699	1.0	1.0
1	17.621	1.0	1.0
2	9.237	1.9	2.0
3	6.164	2.9	3.0
4	4.638	3.8	4.0
8	3.360	5.3	8.0
16	3.308	5.4	16.0

hard.txt			
# threads	**Observed timing**	**Observed speedup compared to original**	**Expected speedup**
original program	5.980	1.0	1.0
1	5.983	1.0	1.0
2	3.146	1.9	2.0
3	2.108	2.8	3.0
4	1.584	3.8	4.0
8	1.136	5.3	8.0
16	1.149	5.2	16.0

hard2.txt			
# threads	**Observed timing**	**Observed speedup compared to original**	**Expected speedup**
original program	5.986	1.0	1.0
1	5.983	1.0	1.0
2	3.146	1.9	2.0
3	2.108	2.8	3.0
4	1.586	3.8	4.0
8	1.131	5.3	8.0
16	0.878	6.8	16.0

The results appear to be similar to what I was expecting which is close (as threads have their own cost to operate / handle) to an N times speedup up to the number of physical CPU cores (in this case 4) and plateauing around the number of CPU threads (in this case 8) onwards. My reasoning for this assumption is that the number of CPU physical cores (4) divided by the total number of threads (8) is the number of hyperthreaded threads (2 per core), which are threads that have to context switch and share the same physical core to be computed (which is expensive and slower than 1 thread per physical core).