**Q1**

The graph below shows the relative speedup of an n-sized thread pool in comparison to a 1-sized thread pool. As you can see, there is a significant decrease in performance for thread pools greater than 1. This may be in part due to the overhead that the threads incur and the way in which tasks are scheduled to threads. From 2 to 4 threads there is a slight increase in performance. This is due to the testing machines quad-core processor, where each thread may be run on a core by itself – thus no thread switching overhead. From 5 to 8 threads there is a decrease in performance, as now there is an added overhead for cores swapping threads in and out.

**Q2**

The graph below shows the relative speedup of n optimistic threads in comparison to 0 optimistic threads. As you can see, the algorithm runs fastest with no optimistic threads, as there is no overhead from thread creation and switching. However there is a relative speedup as the number of optimistic threads used increases from 1 to 7. From 1 to 3 optimistic threads there is a steeper relative speedup than 4 and beyond. This is likely due to the fact that the machine running tests uses a quad-core processor, and 3 optimistic threads implies 4 total running threads. Therefore, each thread may be run on a different core. As the number of optimistic threads increase beyond 3, there is still an overall speedup however less of a difference between them. This is due to the overhead of cores having to swap between threads, because now a core will have 1 or more threads to run.