Name: Cicelia Siu,
NSHE: 5005247749
Assignment 4 Writeup

Note: Times do not include times for user input.
Max Threads I could do is 8.

Parallel Sorting

List Size	2 threads time (ns)	4 threads time (ns)	8 threads time (ns)
16	628558	764379	1037755
32	557030	891798	1293016
64	691059	857471	1454482
128	761858	984178	1557909
256	978782	1214656	1812018

Sequential Sorting w/ std::sort

List Size	1st Try time (ns)	2nd Try time (ns)	3rd Try time (ns)	Average (ns)
16	71722	78696	41544	63987.3
32	72337	74154	73317	73269.3
64	109869	111036	108667	109857.3
128	217869	168575	79833	155425.6
256	298801	305248	432036	345361.6

Conclusion:
Sequential sorting had much faster times in comparison to the Parallel Sorting. This is due to parallel sorting needing to send data into multiple threads and recombining the results.
The times for sequential sorting grew exponentially as list sizes increased. On the other hand, parallel sorting's timers grew more linearly. I would assume from the data that parallel sorting works the better than sequential sorting with large amounts of data.

Question: Hypothetically if we have listSize / 2 amount of threads that could run in parallel, would our parallel sorting algorithm have a better runtime than a sequential sorting algorithm?

Unlikely, unless with large amounts of data. The average time for a sequential sorted list of size 16 is 63987.3 . With 8 threads (half of the list size), the time for parallel sorting is

1037755. This time is on par with list size 64 for sequential sorting. The real times actually seemed better with less threads.