In this problem, you will start by selecting a sorting algorithm to sort 7, 500 random integers with values 1-1, 000, 000. We will use pthreads in the problem to generate a parallel version of your sorting algorithm. If you select code from the web, make sure to cite the source of where you obtained the code. You are also welcome to write your own code from scratch. You will also need to generate the 7, 500 random integers (do not time this part of your program). Make sure to print out the final results of the sorted integers to demonstrate that your code works (do not include a printout of the 7, 500 numbers). Provide your source code in your submission on Canvas.

a. Run your program with 1, 2, 4 and 8 threads on any system of your choosing. The system should have at least 4 coures for you to run your program on. Report the performance of your program.
b. Describe some of the challenges faced when performing sorting with multiple threads.
c. Evaluate both the weak scaling and strong scaling properties of your sorting implementation.

*You should include your C/C++ program for this problem. Also include your written answers to question 2 (a-c) in your homework 1 write-up.

Code adapted from https://www.geeksforgeeks.org/merge-sort-using-multi-threading/
Source Code Name: Question2.cpp
Systems: 16 Cores Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz

a. Performance:
   1 thread – 0.0022s
   2 threads – 0.0017s
   4 threads – 0.0014s
   8 threads – 0.0011s

```
[luo.qiu@ood hw1]$ g++ sort1.cpp -lpthread -o hello
[luo.qiu@ood hw1]$ ./hello
sort use 0.0022 seconds
[luo.qiu@ood hw1]$ g++ sort2.cpp -lpthread -o hello
[luo.qiu@ood hw1]$ ./hello
sort use 0.0017 seconds
[luo.qiu@ood hw1]$ g++ sort4.cpp -lpthread -o hello
[luo.qiu@ood hw1]$ ./hello
sort use 0.0014 seconds
[luo.qiu@ood hw1]$ g++ sort8.cpp -lpthread -o hello
[luo.qiu@ood hw1]$ ./hello
sort use 0.0011 seconds
```

b. Multi-threaded sorting is essentially a process of dividing the entire sorted sequence

into several parts/threads. When the threaded sort completes, it is merged, which is equivalent to merging multiple ordered sequences into one serial sequence. So the problem I faced was to merge the sequences.

c. Evaluation

Strong Scaling: $S = \frac{p}{1+f(p-1)}$

Weak Scaling: $S = p - f(p-1)$

Use strong scaling as the size of the problem has not changed:

	1 thread	2 threads	4 threads	8 threads
S	100%	129.41%	157.14%	200.00%