ECEN 5033 Concurrent Programming Lab 0
Sequential Quicksort Implementation
Sanju Prakash Kannioth

Reason for choosing quicksort:

I Implemented Quicksort mainly because even though both merge sort and quick sort have same time complexity for the average and best case [O(n*logn)] this algorithm is space constant. Quicksort also has better cache locality making it faster in most cases. Also, I have implemented merge sort before but have not implemented quick sort before.

Code organization:

My code has 4 functions apart from the main function – read_sort_write, quicksort, partition and swap.

- The **main** function does the command line argument parsing and calls read_sort_write.
- The **read_sort_write** function is used to read the input file, call the sorting function, write to either stdout or the output file (if specified).
- The **quicksort** function is a recursive function which calls the partition function.
- The **partition** function is used split the array of numbers based on a pivot element. I have chosen the lowest indexed element as the pivot.
- The **swap** function is used to swap elements whenever necessary.

Files submitted:

- **Main.c**: Contains the logic for command line argument parsing and reading and writing to files. Also calls the sorting functions.
- **Quicksort.c**: Contains the function definitions for the functions that are required by the quicksort algorithm.
- **Quicksort.h**: Contains the function declarations for the functions that are required by the quicksort algorithm.
- **Makefile**: Contains targets to build the output file and to clean the build artifacts.
- **Kannioth_SanjuPrakash_Lab0.pdf**: Contains the lab writeup for lab0.

Compilation instructions:

- Run **make** or **make all** to compile the code.
- Run **make clean** to clean the build artifacts and existing text files.

Execution instructions:

The build stage will create an output file named **mysort**.

- Running ***./mysort –name*** will print **Sanju Prakash Kannioth**.
- Running ***./mysort sampleInput.txt*** will sort the numbers in the sampleInput.txt input file and print them to stdout.
- Running ***./mysort sampleInput.txt -o sampleOutput.txt*** will sort the numbers in the sampleInput.txt input file and write the result to sampleOutput.txt output file.