Chris Mathew
Jacek Zarski

Project Part 2:
Multi-Threaded Sorter Documentation & Analysis

1. Design:
   a. We start in main where we initially check the command line args to make sure everything is right in that area
   b. We then go forward and get the real path of the input directory
   c. Use that to begin traversing through the directory recursively
   d. If we stumble on a directory
      i. Create a thread
      ii. We store the path information in a struct for later use
      iii. And we recursively call the traverse function again
   e. If we stumble upon a file
      i. We check if it's a csv if so create a thread
      ii. Then parse the csv
      iii. Check if its valid or not
      iv. Then tokenize and store the csv to a struct
   f. At the end, we do one big merge sort
   g. And then we output the file to one big file
2. Issues
   a. Data Persistence was a huge issue because of the threads sharing memory we had to find an algorithm that would allow each thread to work without interrupting the data writing of another thread
   b. We found that we needed to use mutex locks in order to prevent other threads from writing data to our global struct
   c. Another issue we had was our poor parser/tokenizer algorithm. It would seg fault when there were multiple commas in a line and so we had to fix that in order to get the program to work
   d. We also had to re-figure out mallocing and freeing because there were many time when we were freeing an object we had initially believed was allocated but it turns out it was not
   e. We also had to figure out how to output and create a new file
   f. Lastly, we had to figure out how to put it all together so that it wouldn't crash every time we ran into a lot of CSV files
3. How to run the program
   a. Make sure you have all in the same directory the sorter.c, sorter.h, mergesort.c, and the makefile
   b. In terminal or command prompt run make in the directory of which all of the above is
   c. If that compiles properly, which it should, run make test on the directory of your choosing

d. In the make file, it is imperative you change the directory to be what you need them to be
e. Also make there are no duplicate files, and that all file names have no spaces in them
f. So, for example if you file x.csv then don't have another x.csv
g. Also, no x .csv where there is a space between the x and the "."
h. The directory you want to run the program on should be located where the rest of the files are

Time spent on multi-threaded sorter:

1. 1 csv file: 0.096707
2. 2 csv files: 0.175838
3. 4 csv files: 0.358974
4. 8 csv files: 0.745238
5. 16 csv files: 1.389282
6. 32 csv files: 2.886788
7. 64 csv files: 6.388482
8. 128 csv files: 13.651392
9. 256 csv files: 27.846348
10. 512 csv files: 53. 963518
11. 1024 csv files: 109.794127