Chris Mathew
Jacek Zarski

# Project 2: Multi-Threaded Sorter Analysis

Problems:

1. Timing, we started the project late due to personal reasons as a result we didn't get the best work done as we could.
2. Sharing memory space and writing all the different records to the same file
3. Outputting the file
4. Getting multi-threading to work
5. Make sure the various threads didn't ruin or access old threads memory or work and corrupt it
6. Overwriting the same memory space

Analysis:

1. The multi-process sorter off the bat was very quick
2. I checked speeds on the multi-processor for various amounts of csv files
    a. 2 items: 0.000402 seconds
    b. 4 items: 0.000241 seconds
    c. 8 items: 0.000325 seconds
    d. 16 items: 0.000280 seconds
    e. 32 items: 0.000437 seconds
    f. 64 items: 0.000489 seconds
    g. 128 items: 0.000288 seconds
    h. 256 items: 0.000350 seconds
    i. 512 items: 0.000297 seconds
    j. 1024 items: 0.000469 seconds
3. What I noticed is that the timing has no correlation with the number of csv files, in fact it just had a random time that it took to run indicating that the processing power needed to do this computation is very minimal and hence only took a fraction of the computing power.
4. Unfortunately we were unable to get my multithreaded version to work in time for me to do an analysis thus I would assume that at the very least, the timing would be similar if not almost just as random.

Design:

1. We created a new struct called middleware to house the master record pointer and string for the file path to be passed cause memory of each thread can't be accessed by others