Documentation for Project 2

Sydney Sigue and Yugant Joshi

For this project we decided to start from scratch. We used a lot of code from our previous assignments but it was important to us to check every detail, especially since the code had gotten so much bigger. We decided to first focus on the parameters and making sure we can take in the various ways of how the parameters could be input in ilab. We completed this by taken in each argument and seeing what the argument was after it and that it made sense. Then we focused on getting through all the directories. This part was added in in Project 1. However, for this project it seemed like it would be the first part because parsing. which was the first part in Project 0, would not occur until a csv file was located. This means that first we had to properly move through the directories to find the csv file. When we hit a directory we called pthread_create to create a thread. We decided to created two thread functions, one for when we hit a directory and one for when we hit a csv file. We had a struct to hold the data that we would need to carry into pthread_create. This is something we took time to figure out. We realized there were certain variables that had to be carried into the thread function but because we had multiple variables we didn’t understand how to send those multiple variables into the thread function. We considered making global variables but then we decided it would be better to create a struct. We added in the struct and our thread functions into our header file. After completing this part of the code we added in our parsing code. When we found a csv file it would call its thread function with pthread_create and that function would call parser. In parser we input locks so that two threads would not attempt to write to the same spot in out Records array at the same time. We originally did not do this and we would randomly get segmentation faults in our code, Since the threads are working at the same time in parser we did a pthread_join after parser was done joining all the threads together. Then we print out our thread ids. This is done before mergesort is called. We decided to only do mergesort once. So our code goes through all the directories, finds all the csv files that are set up correctly and it inputs all that data into our Records array. To compile the code you have to type in gcc sorter_thread.c –lpthread. Lpthread is needed to compile code that utilizes threads.