CS4414
Project 2 Maximum Finder
Wright Kim

This project is to find the maximum integer among sequence of integers from the given file. The name of this file should be passed as a command line argument. For my case, after 'make', need to run ./test test.txt in order to read integers from the text file 'test.txt'.

$make

$./test test.txt

Since the maximum number of integers will be 4096, the maximum number threads that will be created will be 2048 and the maximum number of rounds will be 12.

First declared two integer global arrays size of 4096 and initialized them to 0. The reason for two arrays is to write to array2 if read from array1 and vise versa. The array that will be read and write will be swapped for each round and print the final result to the console.

The program will read from the input file and determine the size of an array. If the array size is 0, it means there is no integer, thus, prints to the console that there are no integers. If the size is 1, it is the only integer in the file, thus print to the console.

Need to compare integers if there are more than 2 integers in the file. Inside the while(1), it first checks if the number of threads need to be created is 0 or not. If no more round is needed, the program exits the while loop. Otherwise, it creates multiple threads using for loop. It is possible to create exact amount of threads since we know the number of threads needed, which is half of the size of the number of integer. Here, I have a local integer array which has a value of the index of the for loop. The address of the array will be passed to the pthread_create as an argument so that thread can be use it. Each thread will now be able to access to the global array and pick two integers from it, compare, and write it back to the right location.

In order to make this happen, need to use sem_wait() properly and implement barrier properly. Sem_wait() will block the main thread and wait until each round is finished. The barrier will check a variable count, whether it is 0 or not. If it is not 0, it means that all of the threads have not been done and need to wait with pthread_cond_wait(&cv, &m).

If the count is equal to 0, then the barrier can release all of the threads that has been created by broadcasting. Since the round is over and the main thread can proceed, it also changes the value of the semaphore using sem_post(&s) to unblock the main thread that was blocked by sem_wait().

The barrier function that was implemented must be protected by mutex. If it is not protected by mutex, each thread will try to access the global variable count, which counts whether all of

the threads are finished their work. If it is protected by mutex, only one barrier will able to access ‘count’ at a time, preventing race condition.