Mourya Vulupala (MV638)
Naman Bajaj (NB726)
CS416 – Operating Systems
Project 2

1.
worker_create – This function was sort of the heart of the program (next to the scheduler), as it was where we set up all initial variables and data. The first thing we did was set up the desired thread and all required variables needed for it (stack, context, etc.) This was something that happened everytime the function was called. However, if it was the first time we called worker_create (which we kept track of by checking if the scheduler context was null or not), then we also registered the timer, the scheduler context, and the main thread. After that, we queued the desired thread the user wanted created and depending on if it was the first run, we also added the main thread into the queue, as we would want to return to it to see if the user wanted to add more workers.

worker_yield – This function was relatively straightforward. All we wanted to do was stop the current thread that was running, save its context, push it to the back of the queue and tell the scheduler to pick it up from there. In addition, we also had to do some small tasks such as incrementing the number of context switches and assigning the current thread as null, just to prevent the scheduler from thinking that we are still running the same task.

worker_exit – This was also a pretty straightforward function. We set the required variables of the thread to indicate termination (status, context switches), calculated our average current running average turnaround time, and added the thread to a list of completed threads. We then handed control back to the scheduler.

worker_join – This function relied on the use of worker_yield. Since we had a list that kept track of completed function, we just chose to continuously yield control until the thread that we desired completed running. At which point, we exited the current thread using worker_exit.

worker_mutex_init – This function was relatively small, as we just wanted to allocate space for the mutex the user wanted to initialize, and set its variables to 0.

worker_mutex_lock – In this function, we used __sync_lock_test_and_set to attempt to acquire the mutex. If we were able to do so, we set the mutex's holder to the currently running thread. However, if we were unable to, we pushed the current running thread into a block queue that we set up and set its status accordingly. The block queue just contained threads that were blocked due to being unable to acquire a mutex.

worker_mutex_unlock – In this function, we used __sync_lock_release to released the mutex, we then dequeued from the head of the block list and kept doing so and putting all of the threads into our main queue. This would allow them to compete for the mutex when the scheduler decides it to be appropriate.

worker_mutex_destroy – Destroying the mutex was as simple as just unlocking it and freeing up the memory associated with it.

2. As we increase the worker count for vector_multiply, we notice that the time slowly decreases, but at some points, it hits a limit. We noticed this limit to be around 4 ms (however sometimes it ran it in 2 ms).

3. We find that our library generally runs faster than pthread. We believe that this may be because we don't have nearly the amount of error checking that pthread does and we tend to just assume some things, such as who holds mutex and such that allows us to run faster, but at a greater risk. When considering the pthread library, it needs to be robust and secure, and be general built for a variety of purposes. Our library is more catered around what we think is best and what fits in the terms of this assignment.

4. In terms of collaboration and resources, we used these sources:
- Piazza: Read questions and answers that were posted to get a better understanding of the assignment. Also helped when we were stuck, as many other people got stuck in the same place that we did. This was probably our most helpful and most used resource, just because of the questions that people were asking (we ourselves did not ask any questions).

- General programming resources (GeeksForGeeks, StackOverflow, Google): We used these sources to get a better understanding of the general ideas for the assignment. We did not look at any code in any of these websites. Instead, we used to it understand the conceptual problem, such as how the current threading library (pthread) functions, mutexes, some definitions (we were unsure if Scheduled was the same thing was Running), etc.