Q1 Report
Aum Alok Khatlawala
2020113008

Algorithm Used and Implementation Details:

After receiving the input in the given format, first I assigned the values for arrival time, waiting time and patience time to the struct of each student and created an array of these and sorted it based on the arrival time and in case of a tie in arrival time, based on the index of the student.

After this, I create a semaphore called washing_machine_sem and initialise it to a value of M (number of washing machines) and I initialise an array of threads using the pthread library.

After this, I create the threads one by one and invoke the function for each thread (each student). In the thread function, I implement the following logic:

1. Sleep till arrival time.
2. Do a washing machine semaphore (timed) wait till the patience time of the student.
3. If no washing machine is free by the time the wait operation ends (indicated by an error number called ETIMEDOUT), the student leaves without washing. No sem_post is required here since the semaphore value doesn't decrement in the first place (since no washing machine is allocated).
4. Else, the washing machine is allocated to the student that requires it and we perform a sleep operation for the washing time and then the student leaves after washing. Since this is the critical condition, after this we perform a sem post to free the washing machine.

After the threads finish execution, we join the threads using pthread_join and after this, we destroy the semaphore using sem_destroy.

After this, we print out the statistics using variables that we change during the execution of the thread function.