**Advanced Operating Systems Project-3**

**Objective :** To gain experience with multithreading using the Pthreads library.

We've developed a C program that replicates multiple ticket sellers concurrently selling concert tickets over a span of one hour, utilizing the Pthreads library for thread and mutex creation.

**Overview**: At the program's initiation, there are ten distinct customer queues, each holding N customers. These queues monitor customer ID, their time of arrival, and the duration they're served. Both the arrival and service durations are expressed in minutes, with every customer set to arrive at the commencement of a given minute. Subsequently, every seller receives a customer queue to manage incoming consumers. These sellers are categorized into three types: L, M, and H, which dictates the speed of their service and their seat selection approach. When attending to a customer, a seller identifies an available seat based on their category. A seat becomes available when its associated mutex is not locked. This mechanism ensures that a seat is exclusively reserved by one seller. Upon securing a seat mutex, the seller logs details like response duration, total service time, and the seller's type at that seat. The process continues until either all seats are filled or the one-hour mark is reached. Unserved customers, if any, are informed that no seats are available. After the simulation concludes, metrics such as average response duration, service duration, and throughput for the respective seller types are determined. Shared across functions are variables like total customers for each seller, the overarching clock timer, pthread parameters, and seat-associated data structures.

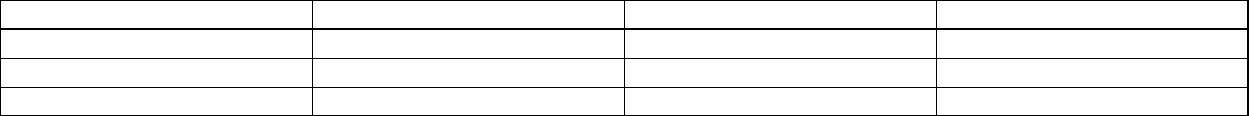
Our model was conceptualized utilizing preliminary code, where we mimicked clock increments in the primary thread while managing critical zones and sales operations in the subsidiary threads that emulate ticket selling activities. The primary thread in our simulation is responsible for generating minute-wise clock ticks.

**Simulation Assumptions**:

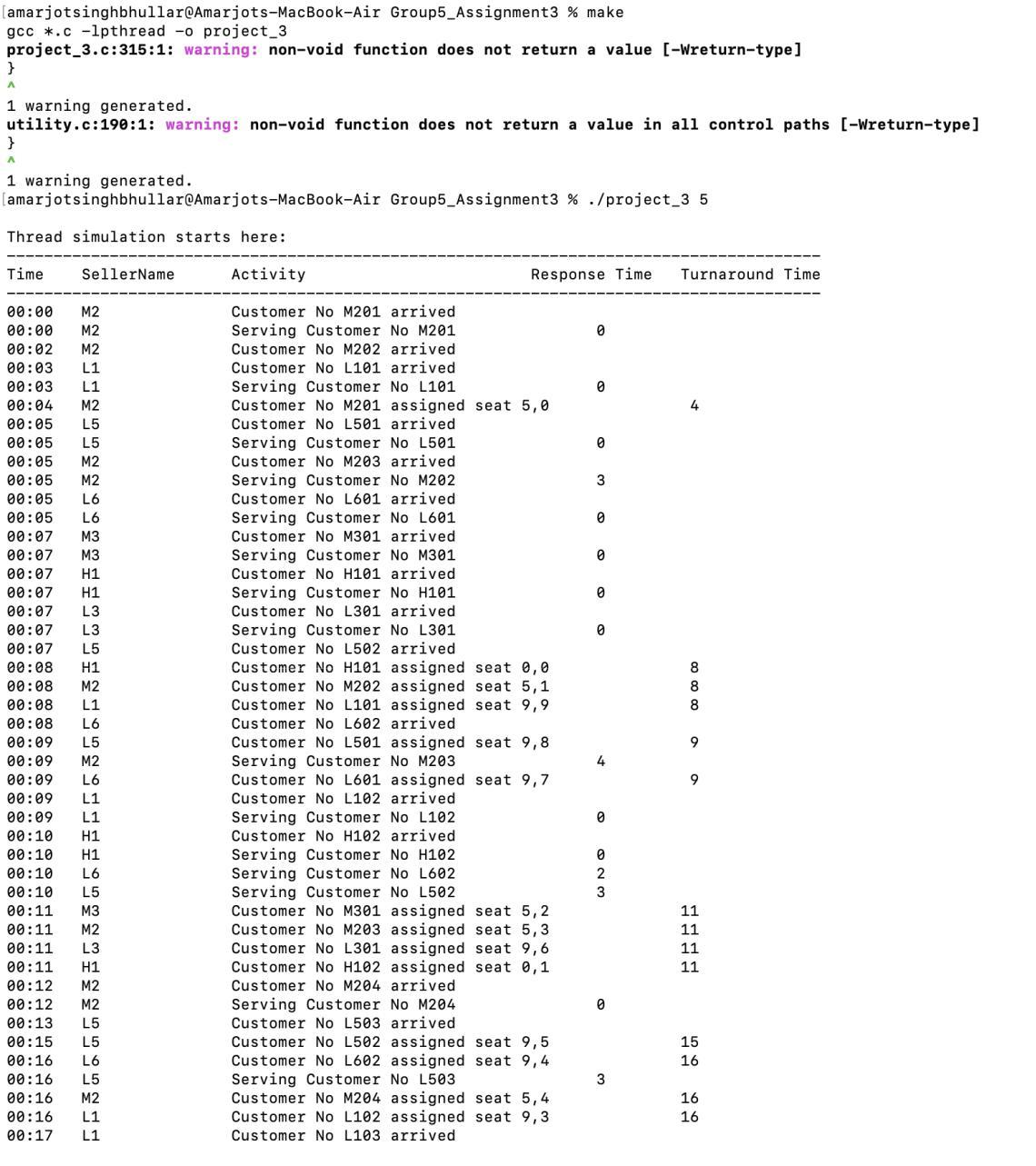
1. Seller Thread Status: At any distinct time, a seller thread could be:
   * Idle: Awaiting the next customer.
   * Attending: Addressing a fresh customer from their queue.
   * Finalizing: Wrapping up the sales procedure for a customer.
   * Operational: Engaging in the actual sales task.
2. Time Increment: The smallest time unit recorded is a minute. Every subsidiary thread symbolizes a minute's activity, like assisting customers or concluding a sale.
3. A fresh clock tick is initiated for maintaining synchronized time.
4. To depict the concert seating, we employed a two-dimensional matrix, presuming that at any given time, only one thread would interact with the matrix, ensuring no clashes in seat allotment.

We also calculated Average Response Time, Average Turnaround Time and Throughput which came out to be:

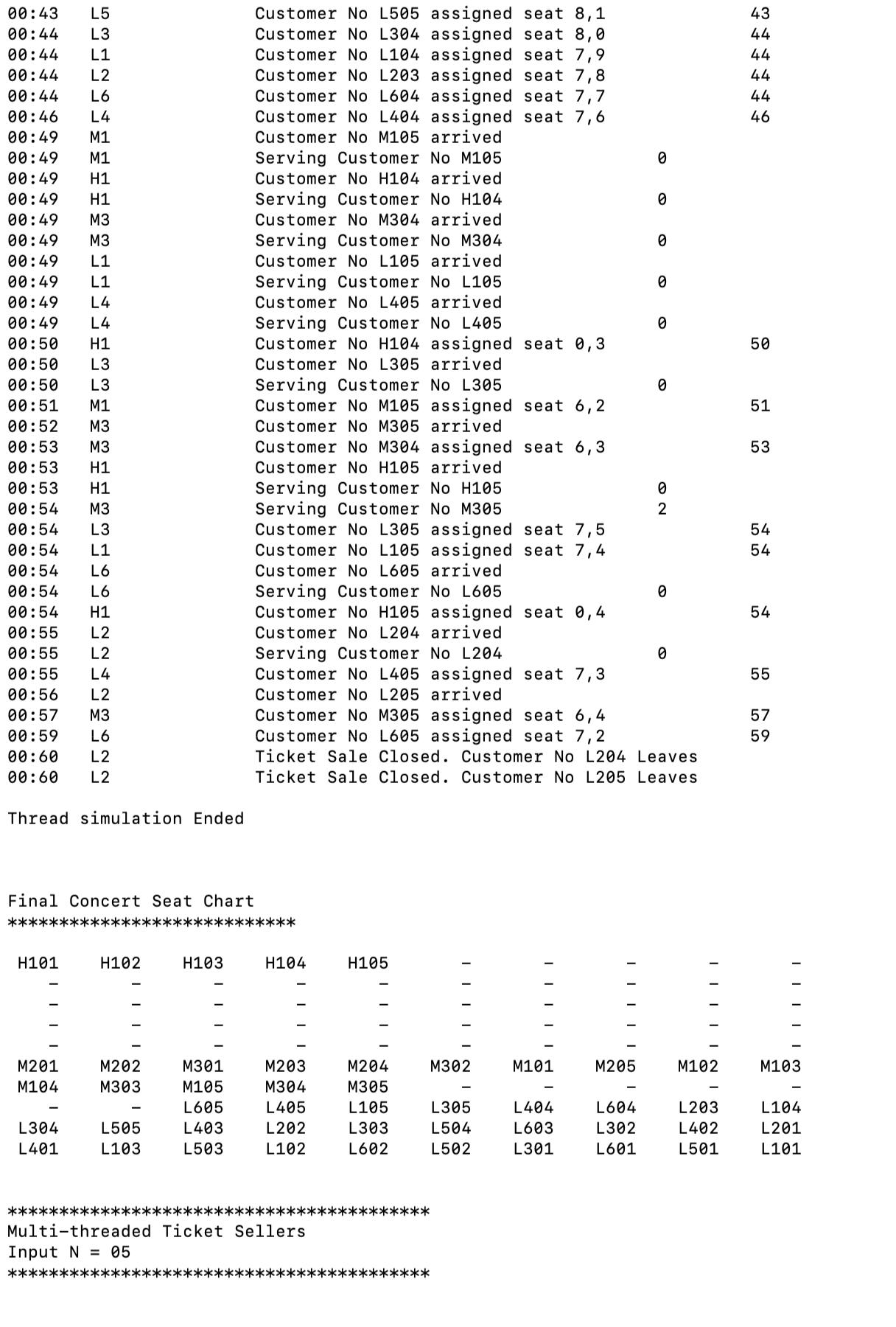
|  | Average Response Time | Average Turnaround Time | Throughput |
| --- | --- | --- | --- |
| H | 0.000000 | 32.60 | 0.08 |
| L | 0.466667 | 30.23 | 0.25 |
| M | 0.800000 | 29.33 | 0.47 |

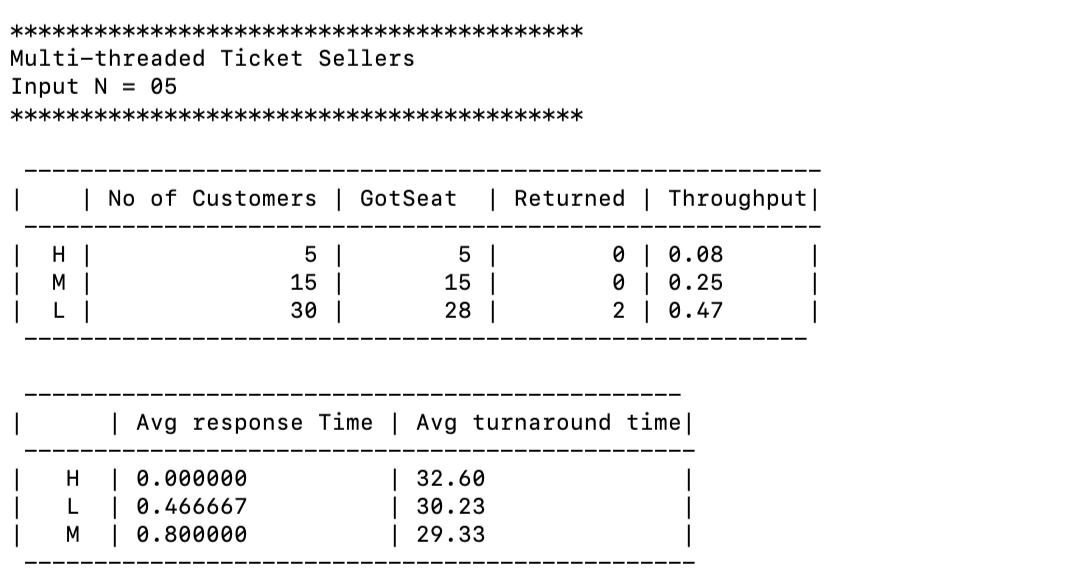


**Output:**









**Conclusion:**

From the results of our simulation, it's clear that the H, M, and L sellers exhibit different response and service durations. The cumulative time required to handle tickets has a cascading effect; it not only extends the service duration for each ticket but also increases the waiting period for every subsequent ticket, thereby affecting both response and service times. Additionally, as more sellers adopt a specific strategy, the average time they spend searching for seats escalates. Given the high number of L sellers, the front rows were rapidly occupied, pushing the L sellers to seek available seats further up. In contrast, the presence of H sellers is dwindling.