Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Report on

"Thread Programming"

[Code No.: COMP 307]

(CE-III/I)

Submitted by

Puspa Hari Ghimire (21)

Group members

Suyog Dhungana (18)

Swochhanda Jangam (25)

Submitted to

Mr. Santosh Khanal

Assistant Professor

Department of Computer Science and Engineering

Submission Date:

June 11, 2024

Simulation of banking model using the concept of thread programming in python.

Source code: https://github.com/puspah-ghimire/Thread-programming

Code Description:

This banking model simulation emulates customer service processes in a bank with three tellers, implementing various CPU scheduling algorithms—FCFS, Non-preemptive SJF, Preemptive SJF, and Round Robin to manage customer queues. Customers arrive at random intervals, each assigned a random service time, and are added to either a regular queue or a priority queue based on the selected algorithm.

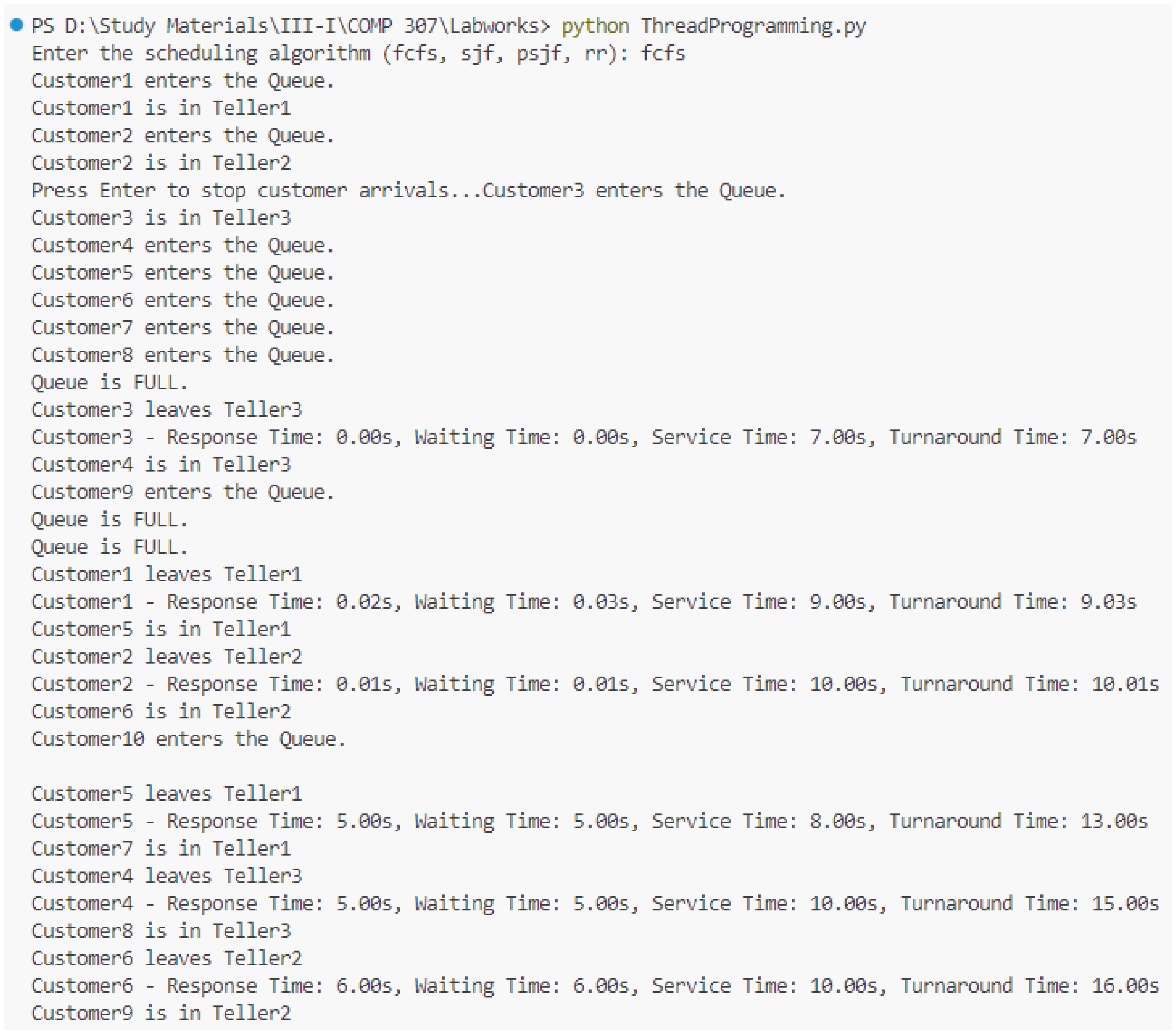
Tellers process customers according to the specified scheduling method, recording service and completion times, and calculating key performance metrics such as response time, waiting time, and turnaround time. The simulation tracks and computes total and average statistics for these metrics. User interaction allows for stopping customer arrivals, and the simulation provides visualizations of teller service times and queue sizes over time.

It offers insights into the performance and behavior of different scheduling algorithms in a multi-teller banking environment.

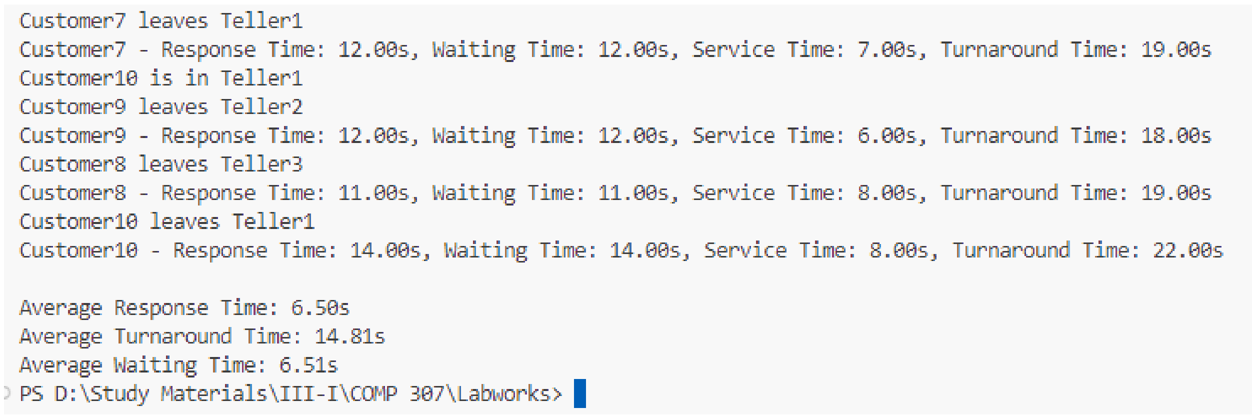
First-Come, First-Served (FCFS):

FCFS is the simplest scheduling algorithm where customers are served in the order they arrive. The teller processes the first customer in the queue until their service is complete before moving on to the next customer. This simple, non-preemptive approach can lead to longer waiting times, especially if a lengthy service precedes shorter ones, causing a "convoy effect" where shorter tasks wait behind longer ones.

Output for FCFS:



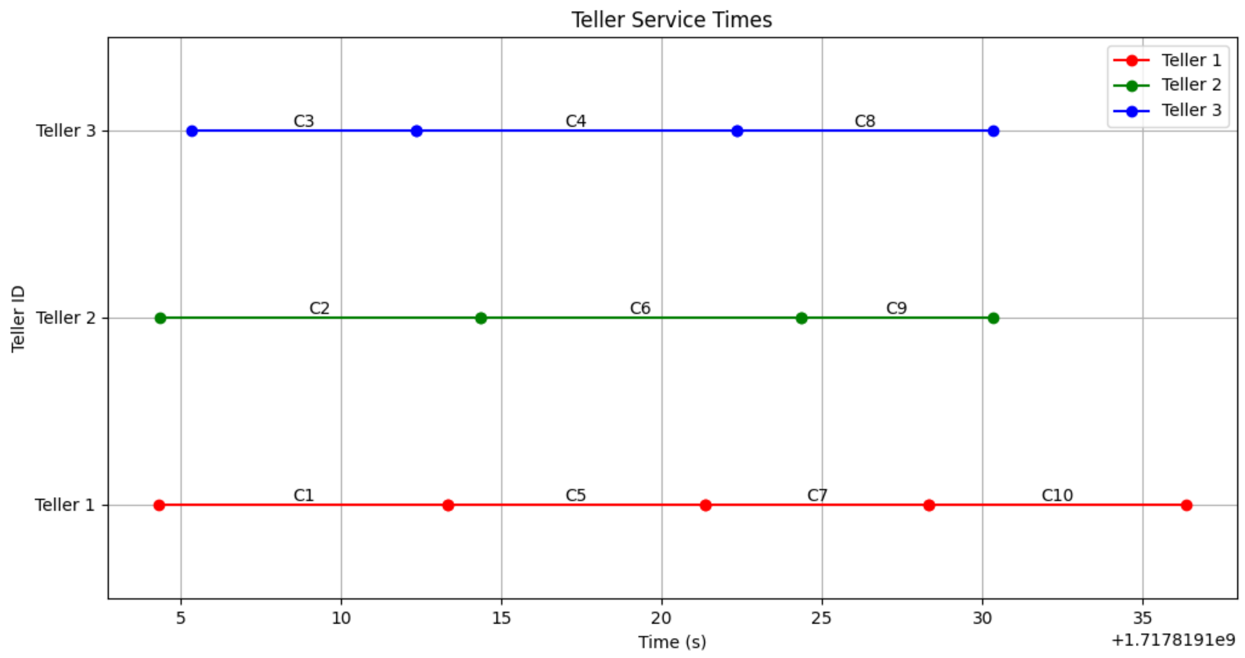
```
PS D:\Study Materials\III-I\COMP 307\Labworks> python ThreadProgramming.py
Enter the scheduling algorithm (fcfs, sjf, psjf, rr): fcfs
Customer1 enters the Queue.
Customer1 is in Teller1
Customer2 enters the Queue.
Customer2 is in Teller2
Press Enter to stop customer arrivals...Customer3 enters the Queue.
Customer3 is in Teller3
Customer4 enters the Queue.
Customer5 enters the Queue.
Customer6 enters the Queue.
Customer7 enters the Queue.
Customer8 enters the Queue.
Queue is FULL.
Customer3 leaves Teller3
Customer3 - Response Time: 0.00s, Waiting Time: 0.00s, Service Time: 7.00s, Turnaround Time: 7.00s
Customer4 is in Teller3
Customer9 enters the Queue.
Queue is FULL.
Queue is FULL.
Customer1 leaves Teller1
Customer1 - Response Time: 0.02s, Waiting Time: 0.03s, Service Time: 9.00s, Turnaround Time: 9.03s
Customer5 is in Teller1
Customer2 leaves Teller2
Customer2 - Response Time: 0.01s, Waiting Time: 0.01s, Service Time: 10.00s, Turnaround Time: 10.01s
Customer6 is in Teller2
Customer10 enters the Queue.

Customer5 leaves Teller1
Customer5 - Response Time: 5.00s, Waiting Time: 5.00s, Service Time: 8.00s, Turnaround Time: 13.00s
Customer7 is in Teller1
Customer4 leaves Teller3
Customer4 - Response Time: 5.00s, Waiting Time: 5.00s, Service Time: 10.00s, Turnaround Time: 15.00s
Customer8 is in Teller3
Customer6 leaves Teller2
Customer6 - Response Time: 6.00s, Waiting Time: 6.00s, Service Time: 10.00s, Turnaround Time: 16.00s
Customer9 is in Teller2
```

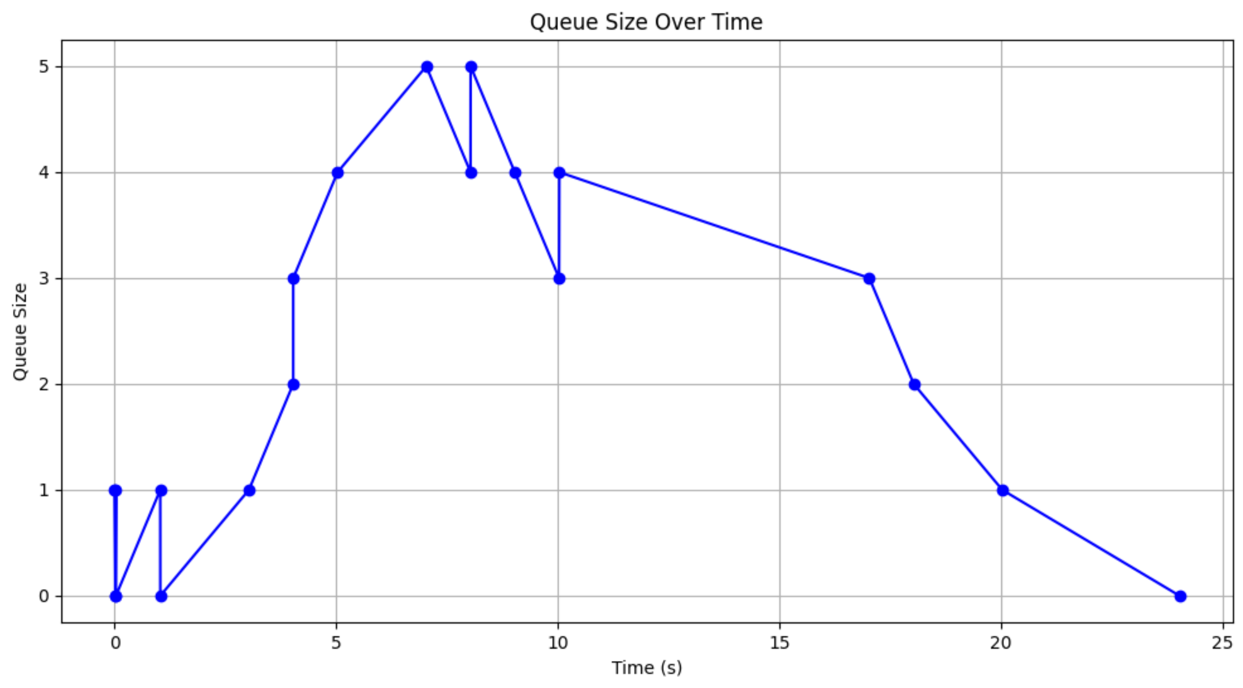


```
Customer7 leaves Teller1
Customer7 - Response Time: 12.00s, Waiting Time: 12.00s, Service Time: 7.00s, Turnaround Time: 19.00s
Customer10 is in Teller1
Customer9 leaves Teller2
Customer9 - Response Time: 12.00s, Waiting Time: 12.00s, Service Time: 6.00s, Turnaround Time: 18.00s
Customer8 leaves Teller3
Customer8 - Response Time: 11.00s, Waiting Time: 11.00s, Service Time: 8.00s, Turnaround Time: 19.00s
Customer10 leaves Teller1
Customer10 - Response Time: 14.00s, Waiting Time: 14.00s, Service Time: 8.00s, Turnaround Time: 22.00s

Average Response Time: 6.50s
Average Turnaround Time: 14.81s
Average Waiting Time: 6.51s
PS D:\Study Materials\III-I\COMP 307\Labworks>
```

The average response time and average waiting time are the same for FCFS scheduling.

Graph for FCFS:



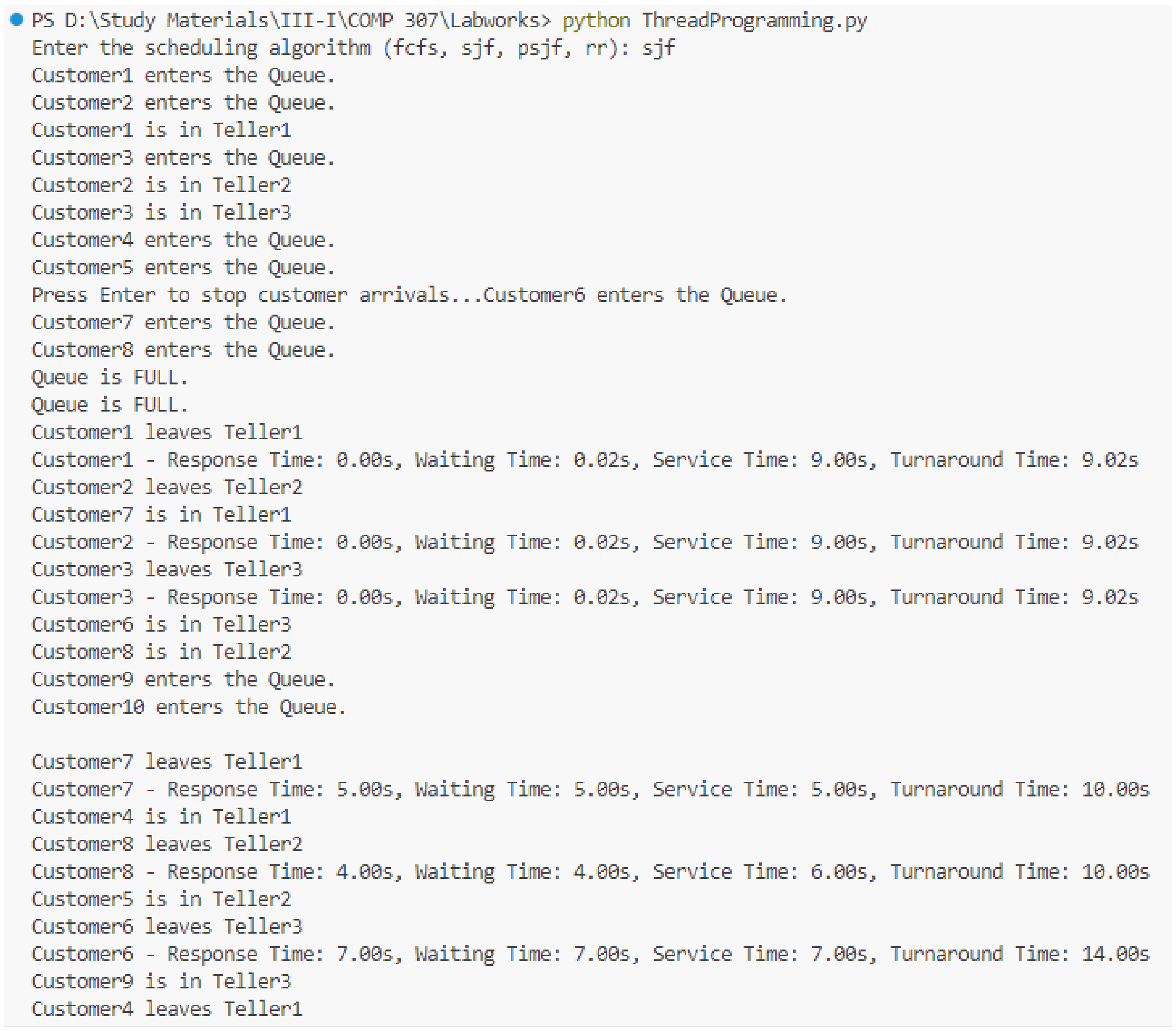


The first graph shows the service times of each customer in their respective teller. C1, C2, C3… are customers and they arrive according to their id in the regular queue. The customers are served sequentially according to their arrival time. The second graph represents the size of the regular queue with respect to time.

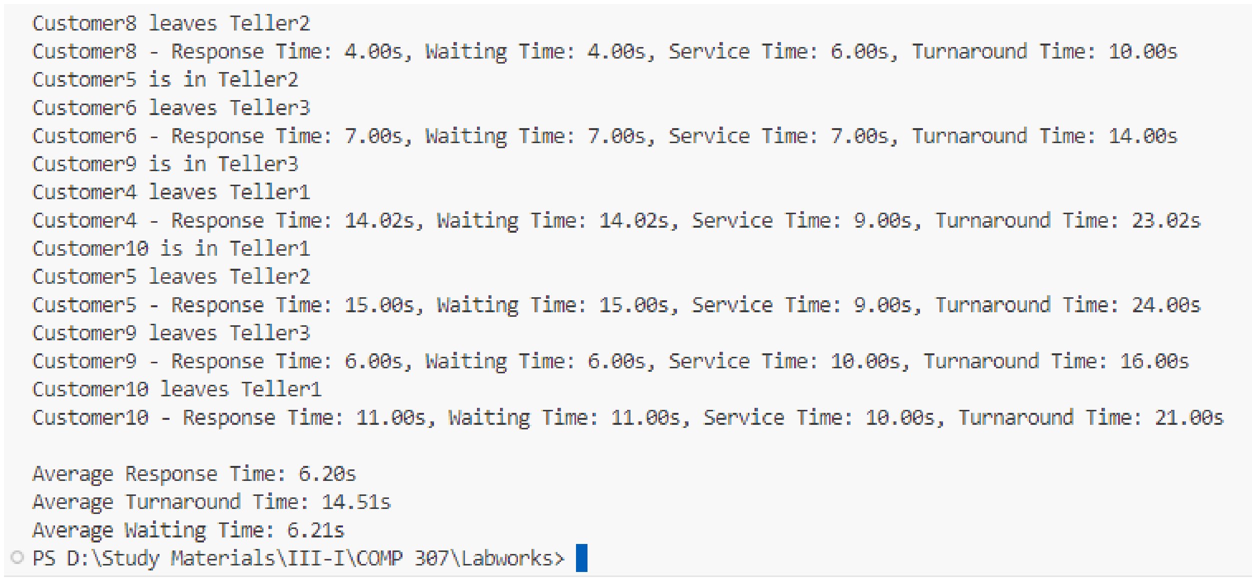
Non-preemptive Shortest Job First (SJF):

Here, customers are prioritized based on their service time, with the shortest jobs being served first. Customers are added to a priority queue sorted by service time. Since it is a non-preemptive method, once the customer starts being served, it leaves the teller only after the completion of its service time. This approach minimizes average waiting time but can cause longer jobs to starve if shorter ones keep arriving.

Output for SJF:

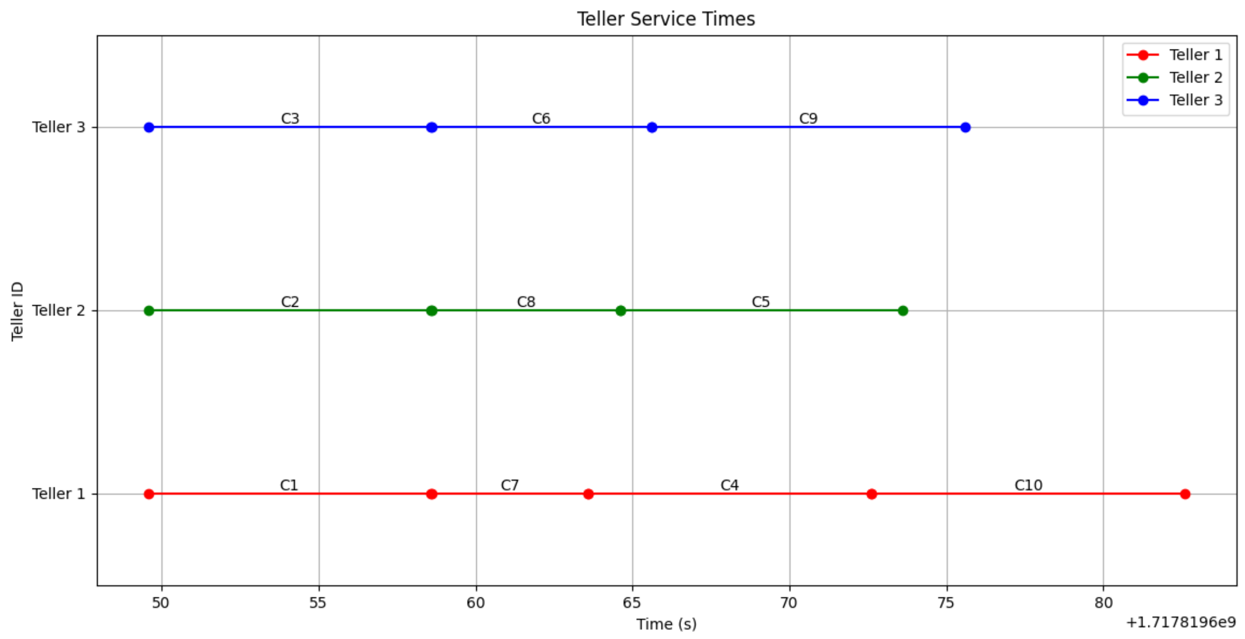


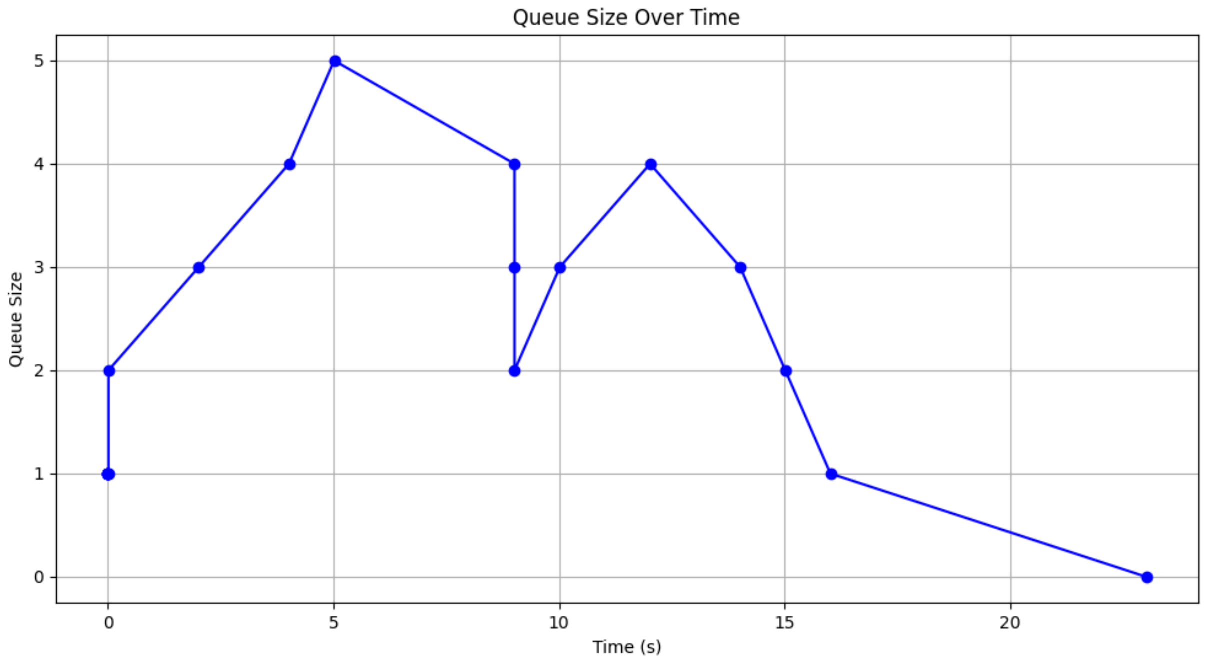
```
PS D:\Study Materials\III-I\COMP 307\Labworks> python ThreadProgramming.py
Enter the scheduling algorithm (fcfs, sjf, psjf, rr): sjf
Customer1 enters the Queue.
Customer2 enters the Queue.
Customer1 is in Teller1
Customer3 enters the Queue.
Customer2 is in Teller2
Customer3 is in Teller3
Customer4 enters the Queue.
Customer5 enters the Queue.
Press Enter to stop customer arrivals...Customer6 enters the Queue.
Customer7 enters the Queue.
Customer8 enters the Queue.
Queue is FULL.
Queue is FULL.
Customer1 leaves Teller1
Customer1 - Response Time: 0.00s, Waiting Time: 0.02s, Service Time: 9.00s, Turnaround Time: 9.02s
Customer2 leaves Teller2
Customer7 is in Teller1
Customer2 - Response Time: 0.00s, Waiting Time: 0.02s, Service Time: 9.00s, Turnaround Time: 9.02s
Customer3 leaves Teller3
Customer3 - Response Time: 0.00s, Waiting Time: 0.02s, Service Time: 9.00s, Turnaround Time: 9.02s
Customer6 is in Teller3
Customer8 is in Teller2
Customer9 enters the Queue.
Customer10 enters the Queue.

Customer7 leaves Teller1
Customer7 - Response Time: 5.00s, Waiting Time: 5.00s, Service Time: 5.00s, Turnaround Time: 10.00s
Customer4 is in Teller1
Customer8 leaves Teller2
Customer8 - Response Time: 4.00s, Waiting Time: 4.00s, Service Time: 6.00s, Turnaround Time: 10.00s
Customer5 is in Teller2
Customer6 leaves Teller3
Customer6 - Response Time: 7.00s, Waiting Time: 7.00s, Service Time: 7.00s, Turnaround Time: 14.00s
Customer9 is in Teller3
Customer4 leaves Teller1
```



```
Customer8 leaves Teller2
Customer8 - Response Time: 4.00s, Waiting Time: 4.00s, Service Time: 6.00s, Turnaround Time: 10.00s
Customer5 is in Teller2
Customer6 leaves Teller3
Customer6 - Response Time: 7.00s, Waiting Time: 7.00s, Service Time: 7.00s, Turnaround Time: 14.00s
Customer9 is in Teller3
Customer4 leaves Teller1
Customer4 - Response Time: 14.02s, Waiting Time: 14.02s, Service Time: 9.00s, Turnaround Time: 23.02s
Customer10 is in Teller1
Customer5 leaves Teller2
Customer5 - Response Time: 15.00s, Waiting Time: 15.00s, Service Time: 9.00s, Turnaround Time: 24.00s
Customer9 leaves Teller3
Customer9 - Response Time: 6.00s, Waiting Time: 6.00s, Service Time: 10.00s, Turnaround Time: 16.00s
Customer10 leaves Teller1
Customer10 - Response Time: 11.00s, Waiting Time: 11.00s, Service Time: 10.00s, Turnaround Time: 21.00s

Average Response Time: 6.20s
Average Turnaround Time: 14.51s
Average Waiting Time: 6.21s
PS D:\Study Materials\III-I\COMP 307\Labworks>
```

Graph for SJF:



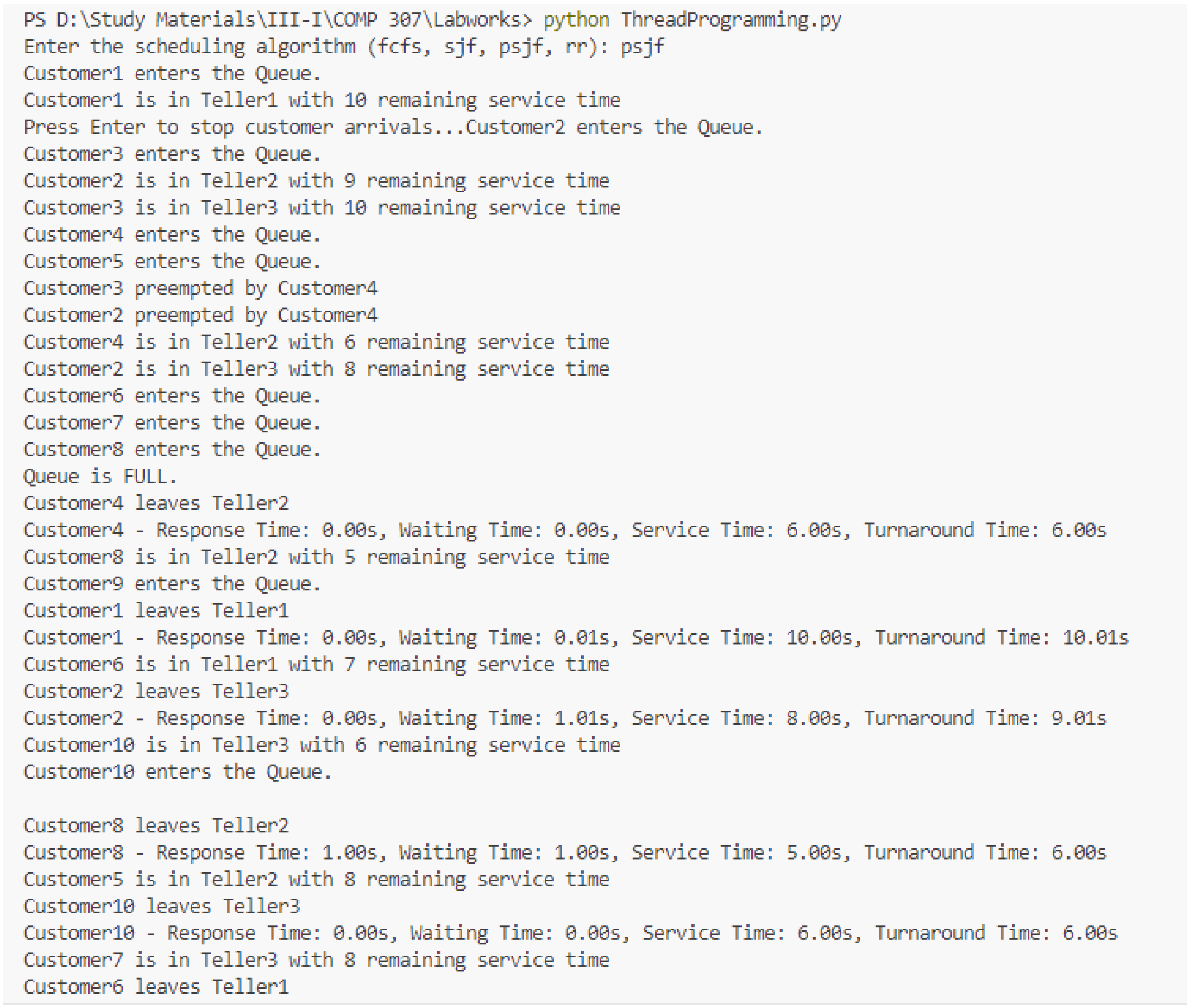


The first graph shows the service times of each customer in their respective teller. C1, C2, C3… are customers arriving in the priority queue where customer having shortest service time is placed at the front of the queue. We observe that although C7 is arrived later than C4, it is being served first due to shorter service time. The second graph represents the size of the priority queue with respect to time.

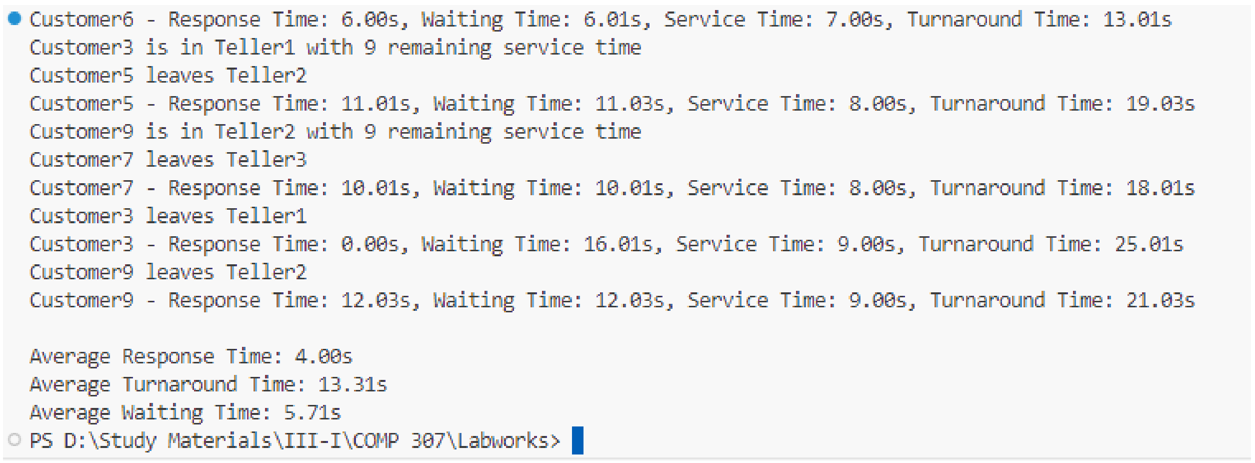
Preemptive Shortest Job First (PSJF) (aka Shortest Remaining Time First):

This algorithm is a preemptive version of SJF. If a new customer arrives with a shorter remaining service time than the current customer being served, the current service is interrupted, and the new customer is served. This ensures the shortest jobs are always prioritized, further minimizing waiting and turnaround times. However, it introduces complexity due to frequent context switching.

Output for PSJF:

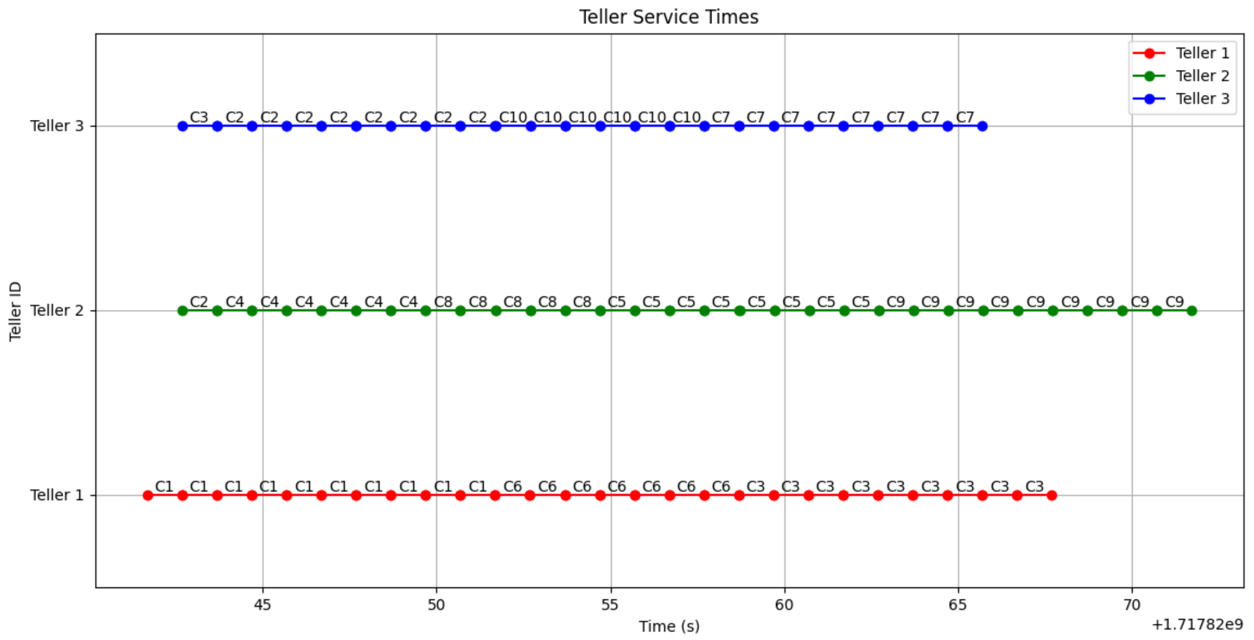


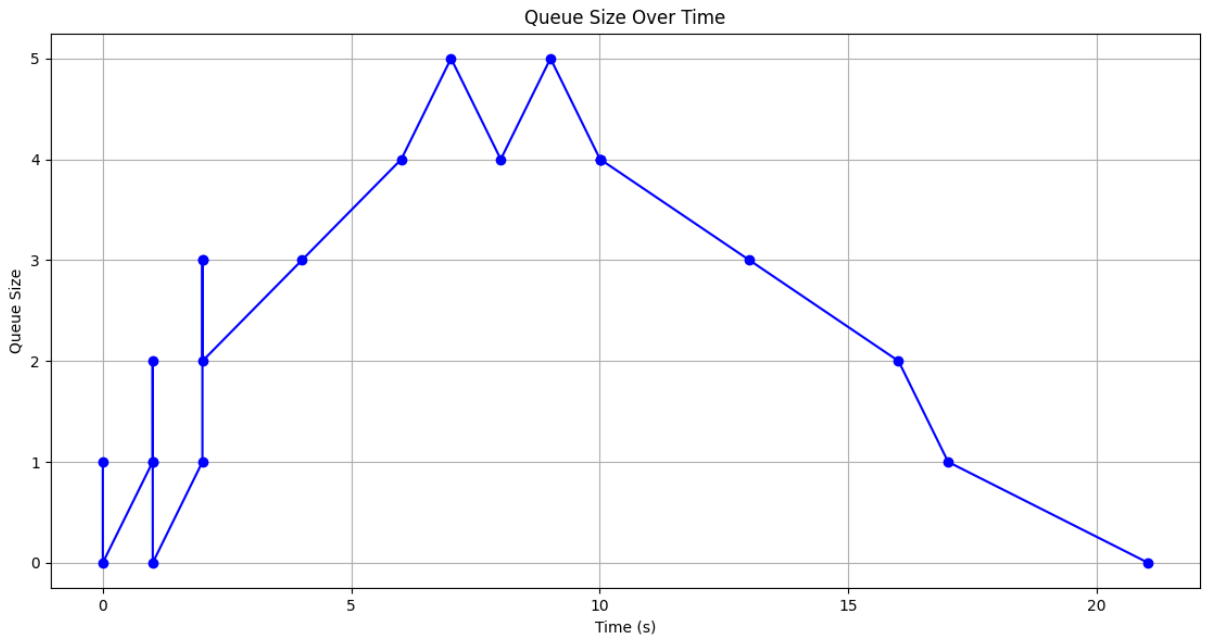
```
PS D:\Study Materials\III-I\COMP 307\Labworks> python ThreadProgramming.py
Enter the scheduling algorithm (fcfs, sjf, psjf, rr): psjf
Customer1 enters the Queue.
Customer1 is in Teller1 with 10 remaining service time
Press Enter to stop customer arrivals...Customer2 enters the Queue.
Customer3 enters the Queue.
Customer2 is in Teller2 with 9 remaining service time
Customer3 is in Teller3 with 10 remaining service time
Customer4 enters the Queue.
Customer5 enters the Queue.
Customer3 preempted by Customer4
Customer2 preempted by Customer4
Customer4 is in Teller2 with 6 remaining service time
Customer2 is in Teller3 with 8 remaining service time
Customer6 enters the Queue.
Customer7 enters the Queue.
Customer8 enters the Queue.
Queue is FULL.
Customer4 leaves Teller2
Customer4 - Response Time: 0.00s, Waiting Time: 0.00s, Service Time: 6.00s, Turnaround Time: 6.00s
Customer8 is in Teller2 with 5 remaining service time
Customer9 enters the Queue.
Customer1 leaves Teller1
Customer1 - Response Time: 0.00s, Waiting Time: 0.01s, Service Time: 10.00s, Turnaround Time: 10.01s
Customer6 is in Teller1 with 7 remaining service time
Customer2 leaves Teller3
Customer2 - Response Time: 0.00s, Waiting Time: 1.01s, Service Time: 8.00s, Turnaround Time: 9.01s
Customer10 is in Teller3 with 6 remaining service time
Customer10 enters the Queue.

Customer8 leaves Teller2
Customer8 - Response Time: 1.00s, Waiting Time: 1.00s, Service Time: 5.00s, Turnaround Time: 6.00s
Customer5 is in Teller2 with 8 remaining service time
Customer10 leaves Teller3
Customer10 - Response Time: 0.00s, Waiting Time: 0.00s, Service Time: 6.00s, Turnaround Time: 6.00s
Customer7 is in Teller3 with 8 remaining service time
Customer6 leaves Teller1
```



```
Customer6 - Response Time: 6.00s, Waiting Time: 6.01s, Service Time: 7.00s, Turnaround Time: 13.01s
Customer3 is in Teller1 with 9 remaining service time
Customer5 leaves Teller2
Customer5 - Response Time: 11.01s, Waiting Time: 11.03s, Service Time: 8.00s, Turnaround Time: 19.03s
Customer9 is in Teller2 with 9 remaining service time
Customer7 leaves Teller3
Customer7 - Response Time: 10.01s, Waiting Time: 10.01s, Service Time: 8.00s, Turnaround Time: 18.01s
Customer3 leaves Teller1
Customer3 - Response Time: 0.00s, Waiting Time: 16.01s, Service Time: 9.00s, Turnaround Time: 25.01s
Customer9 leaves Teller2
Customer9 - Response Time: 12.03s, Waiting Time: 12.03s, Service Time: 9.00s, Turnaround Time: 21.03s

Average Response Time: 4.00s
Average Turnaround Time: 13.31s
Average Waiting Time: 5.71s
PS D:\Study Materials\III-I\COMP 307\Labworks>
```

Graph for PSJF:

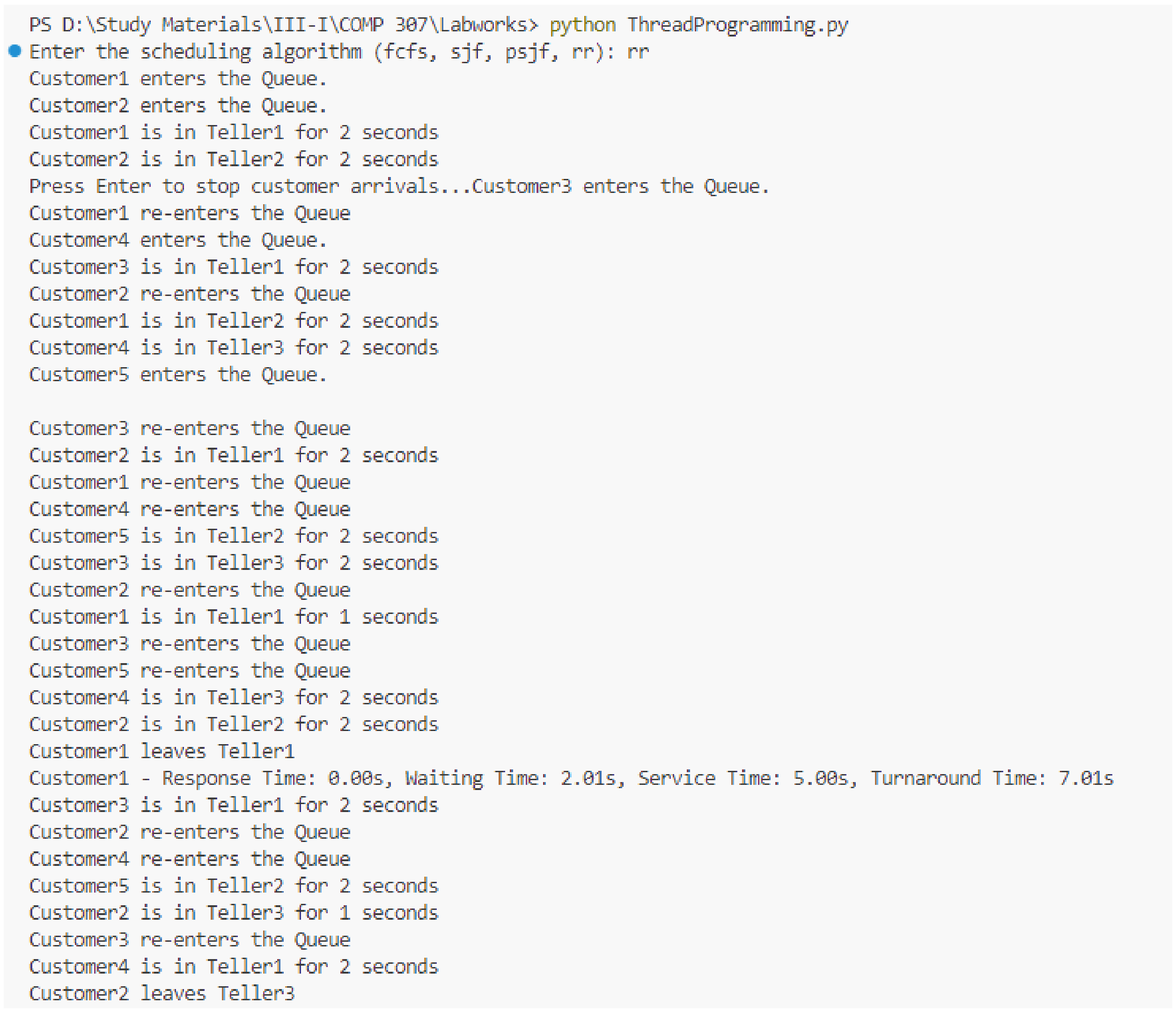




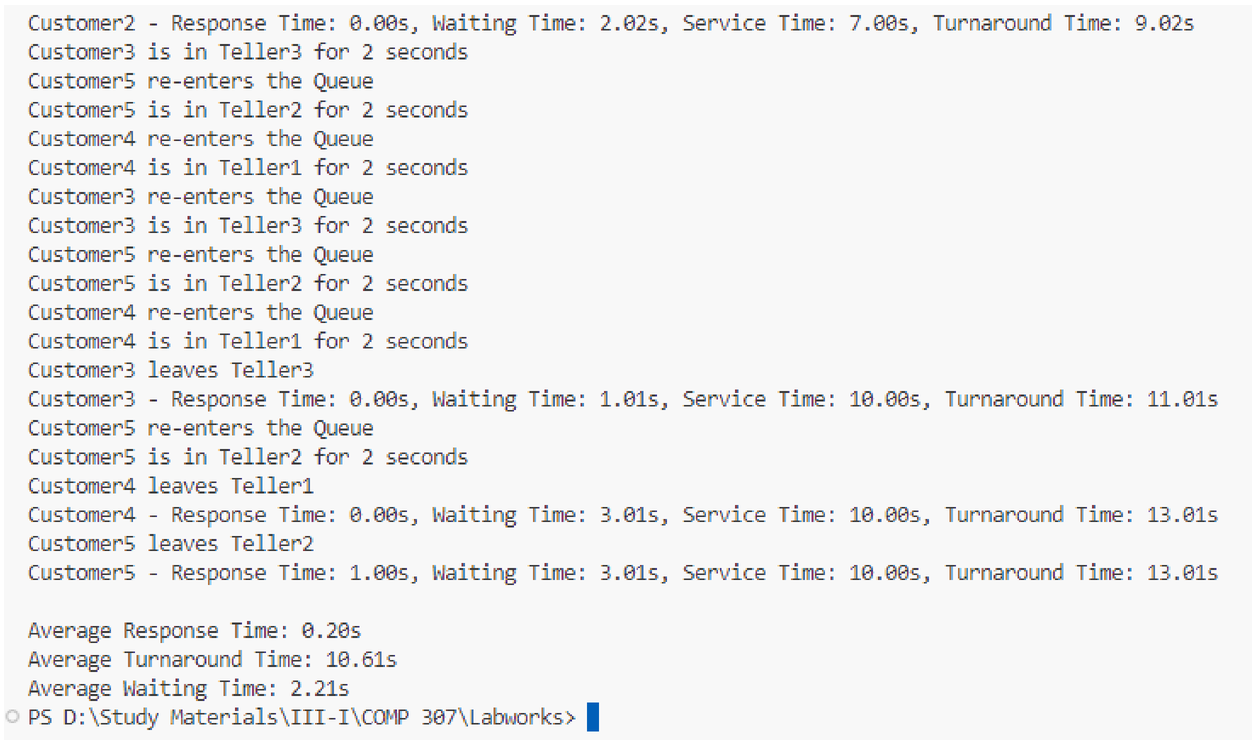
The first graph shows the service times of each customer in their respective teller. C1, C2, C3… are customers and they arrive according to their id in the priority queue where customer having shortest remaining service time is placed at the front of the queue. The algorithm checks for the shortest remaining time in the queue in every 1 second. In the graph, we see that C2 with 7s of remaining service time is preempted by C4 having 6s of service time.
The second graph represents the size of the priority queue with respect to time. The sudden spikes represent the change in the size of queue instantaneously.

Round Robin (RR):
Round Robin scheduling assigns a fixed time quantum (here 2 seconds) for each customer in the queue. Customers are served in cyclic order for a set time slice. If a customer's service isn't complete within the time quantum, they are placed back in the queue to wait for another turn. This method ensures fair and equitable service distribution, reducing response time but potentially increasing context switching overhead if the time quantum is too small.

Output for RR:



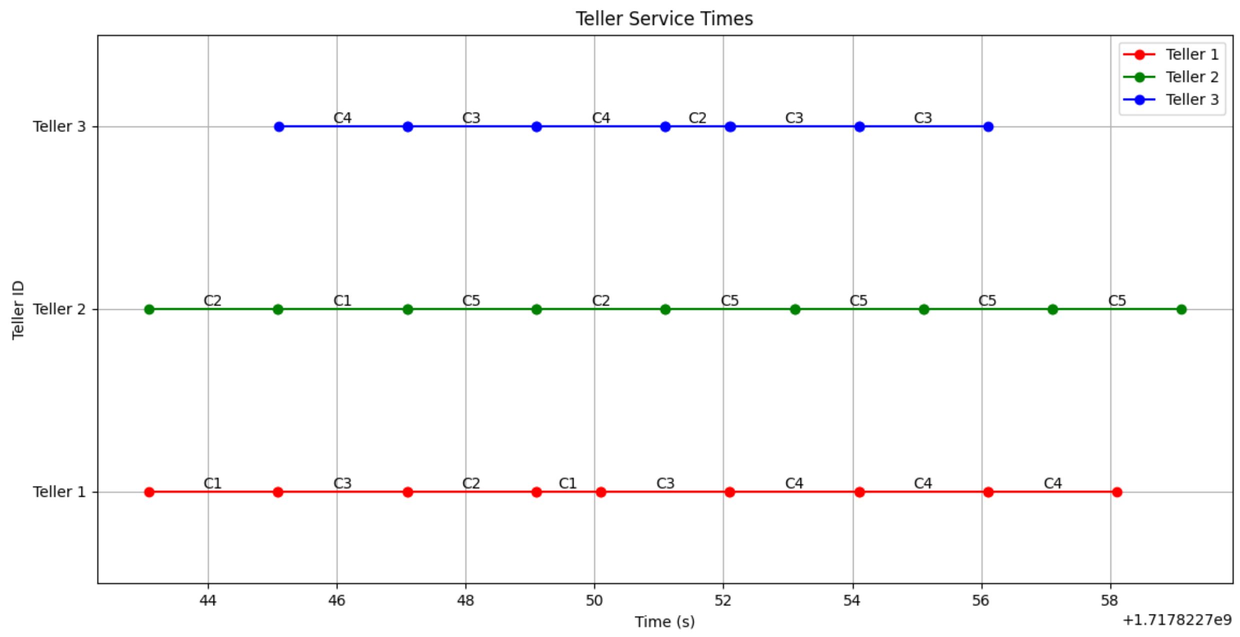
```
PS D:\Study Materials\III-I\COMP 307\Labworks> python ThreadProgramming.py
Enter the scheduling algorithm (fcfs, sjf, psjf, rr): rr
Customer1 enters the Queue.
Customer2 enters the Queue.
Customer1 is in Teller1 for 2 seconds
Customer2 is in Teller2 for 2 seconds
Press Enter to stop customer arrivals...Customer3 enters the Queue.
Customer1 re-enters the Queue
Customer4 enters the Queue.
Customer3 is in Teller1 for 2 seconds
Customer2 re-enters the Queue
Customer1 is in Teller2 for 2 seconds
Customer4 is in Teller3 for 2 seconds
Customer5 enters the Queue.

Customer3 re-enters the Queue
Customer2 is in Teller1 for 2 seconds
Customer1 re-enters the Queue
Customer4 re-enters the Queue
Customer5 is in Teller2 for 2 seconds
Customer3 is in Teller3 for 2 seconds
Customer2 re-enters the Queue
Customer1 is in Teller1 for 1 seconds
Customer3 re-enters the Queue
Customer5 re-enters the Queue
Customer4 is in Teller3 for 2 seconds
Customer2 is in Teller2 for 2 seconds
Customer1 leaves Teller1
Customer1 - Response Time: 0.00s, Waiting Time: 2.01s, Service Time: 5.00s, Turnaround Time: 7.01s
Customer3 is in Teller1 for 2 seconds
Customer2 re-enters the Queue
Customer4 re-enters the Queue
Customer5 is in Teller2 for 2 seconds
Customer2 is in Teller3 for 1 seconds
Customer3 re-enters the Queue
Customer4 is in Teller1 for 2 seconds
Customer2 leaves Teller3
```

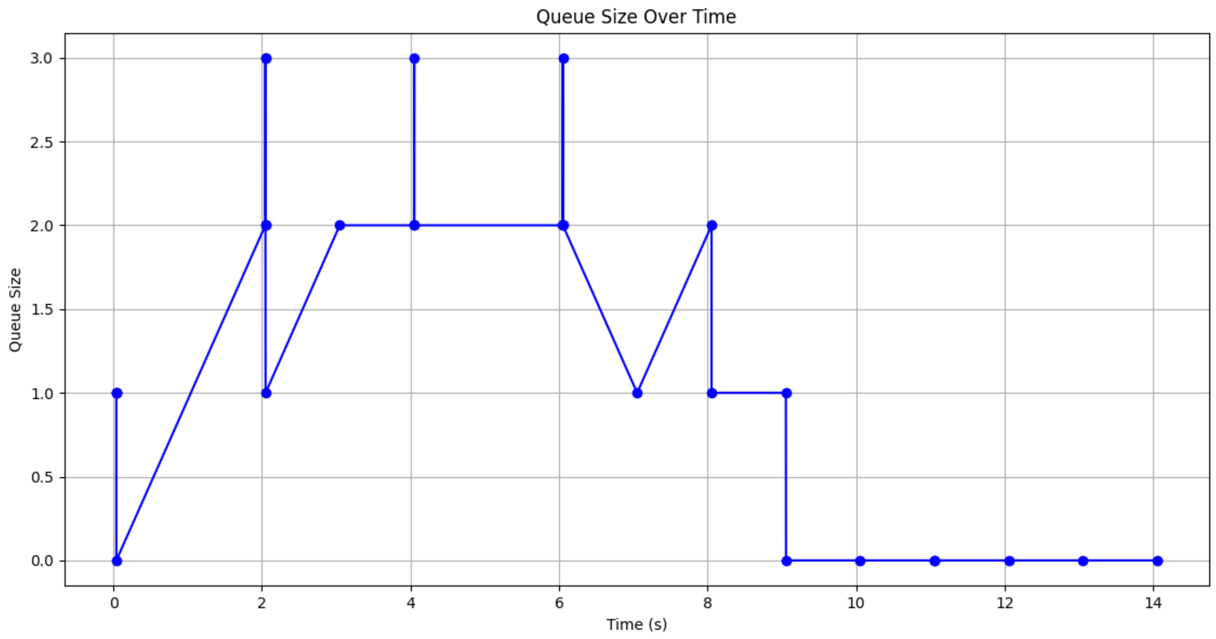


```
Customer2 - Response Time: 0.00s, Waiting Time: 2.02s, Service Time: 7.00s, Turnaround Time: 9.02s
Customer3 is in Teller3 for 2 seconds
Customer5 re-enters the Queue
Customer5 is in Teller2 for 2 seconds
Customer4 re-enters the Queue
Customer4 is in Teller1 for 2 seconds
Customer3 re-enters the Queue
Customer3 is in Teller3 for 2 seconds
Customer5 re-enters the Queue
Customer5 is in Teller2 for 2 seconds
Customer4 re-enters the Queue
Customer4 is in Teller1 for 2 seconds
Customer3 leaves Teller3
Customer3 - Response Time: 0.00s, Waiting Time: 1.01s, Service Time: 10.00s, Turnaround Time: 11.01s
Customer5 re-enters the Queue
Customer5 is in Teller2 for 2 seconds
Customer4 leaves Teller1
Customer4 - Response Time: 0.00s, Waiting Time: 3.01s, Service Time: 10.00s, Turnaround Time: 13.01s
Customer5 leaves Teller2
Customer5 - Response Time: 1.00s, Waiting Time: 3.01s, Service Time: 10.00s, Turnaround Time: 13.01s

Average Response Time: 0.20s
Average Turnaround Time: 10.61s
Average Waiting Time: 2.21s
PS D:\Study Materials\III-I\COMP 307\Labworks>
```

It has the lowest average response time of all algorithms.

Graph for RR:





The first graph shows the service times of each customer in their respective teller. C1, C2, C3… are customers and they arrive according to their id in the regular queue. We clearly observe that after every 2 seconds, the customers are placed back into the queue and served in a cyclic manner.

The second graph represents the size of the regular queue with respect to time. The sudden spikes represent the change in the size of queue instantaneously.

Comparison Chart:

Algorithm	FCFS	Non-preemptive SJF	Preemptive SJF	RR
Average Response Time	Generally high, can be high for longer jobs	Typically, lower than FCFS, depends on job length distribution	Lower than non-preemptive SJF for varying job lengths	Can be low, depends on time quantum
Average Waiting Time	Can be high, especially if long jobs arrive first	Lower than FCFS, better with shorter jobs first	Typically, the lowest, as jobs are processed more fairly	Can be higher than SJF, but better with smaller time quantum
Average Turnaround Time	High for longer jobs arriving first	Lower than FCFS, better with shorter jobs	Generally lower, due to preemption	Depends on time quantum, balances between waiting and turnaround

Conclusion:

Each algorithm has its advantages and disadvantages, and the choice of an appropriate scheduling algorithm should be guided by the specific requirements of the application, such as the need for fairness, efficiency, response time, and the ability to handle varied job lengths effectively.