Palestine Polytechnic University

Operating System

Course Project :

Process Scheduling Simulation Program

Done By :

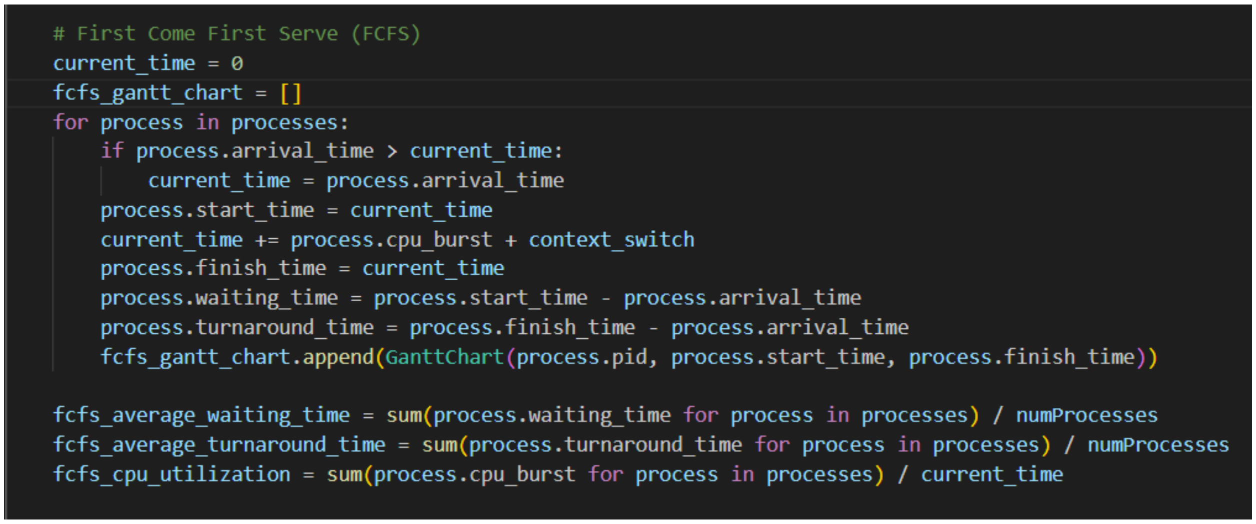
Mohammed Muamar (211028)

Supervisor:

Dr. Hani Salah

Simulates the scheduling of processes using First-Come First-Served (FCFS)

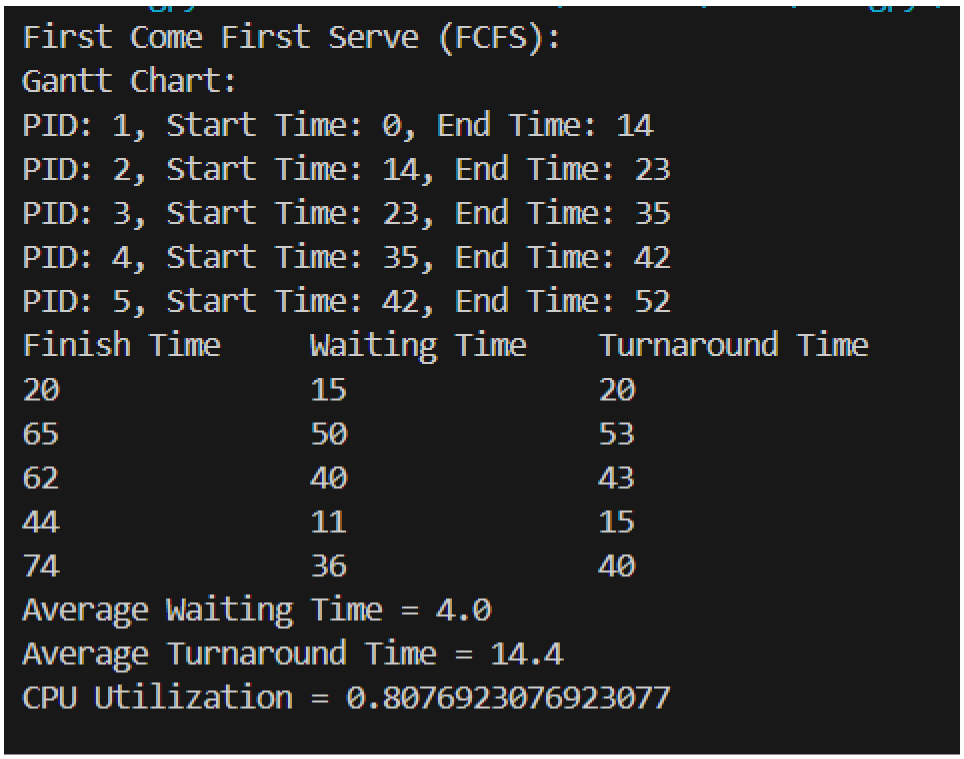
Python code :



```python
# First Come First Serve (FCFS)
current_time = 0
fcfs_gantt_chart = []
for process in processes:
    if process.arrival_time > current_time:
        current_time = process.arrival_time
    process.start_time = current_time
    current_time += process.cpu_burst + context_switch
    process.finish_time = current_time
    process.waiting_time = process.start_time - process.arrival_time
    process.turnaround_time = process.finish_time - process.arrival_time
    fcfs_gantt_chart.append(GanttChart(process.pid, process.start_time, process.finish_time))

fcfs_average_waiting_time = sum(process.waiting_time for process in processes) / numProcesses
fcfs_average_turnaround_time = sum(process.turnaround_time for process in processes) / numProcesses
fcfs_cpu_utilization = sum(process.cpu_burst for process in processes) / current_time
```

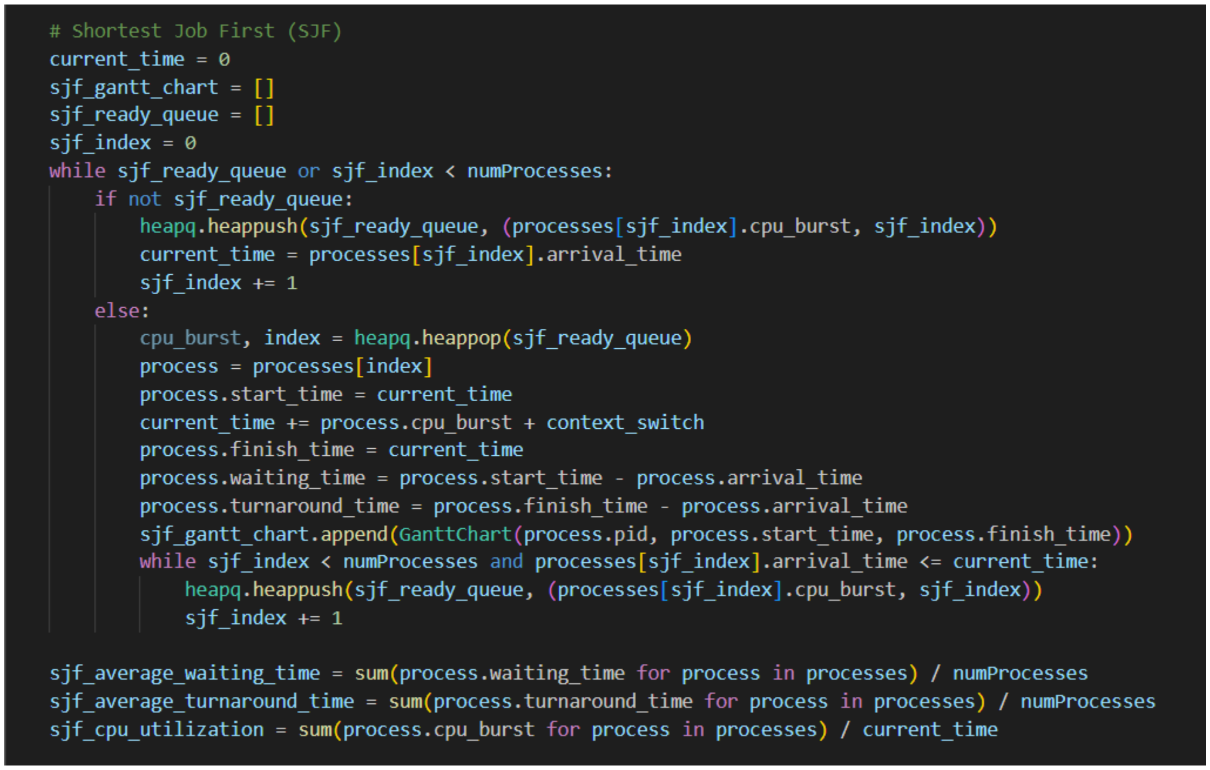
Code output :



```
First Come First Serve (FCFS):
Gantt Chart:
PID: 1, Start Time: 0, End Time: 14
PID: 2, Start Time: 14, End Time: 23
PID: 3, Start Time: 23, End Time: 35
PID: 4, Start Time: 35, End Time: 42
PID: 5, Start Time: 42, End Time: 52
Finish Time     Waiting Time     Turnaround Time
20              15               20
65              50               53
62              40               43
44              11               15
74              36               40
Average Waiting Time = 4.0
Average Turnaround Time = 14.4
CPU Utilization = 0.8076923076923077
```

Simulates the scheduling of processes using Shortest Remaining Time (SRT)

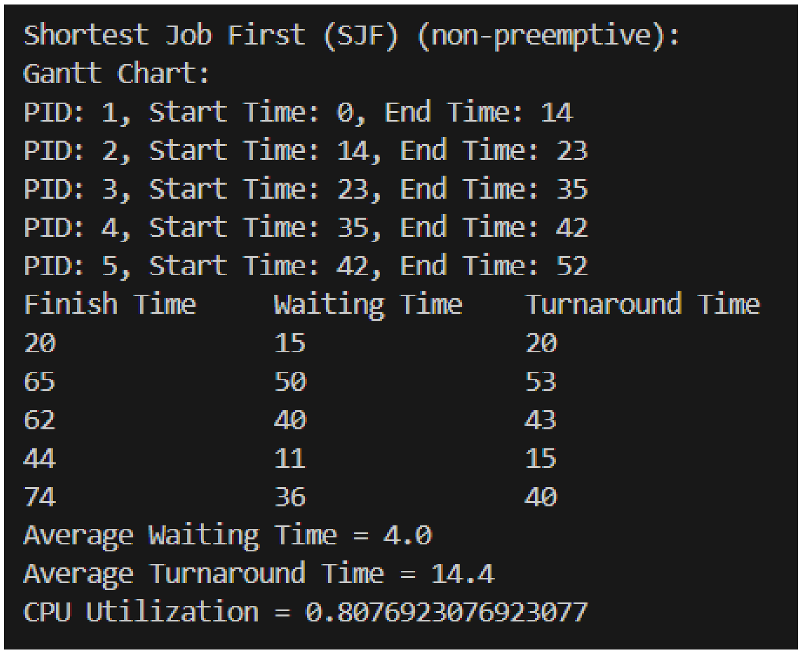
Python code :



```python
# Shortest Job First (SJF)
current_time = 0
sjf_gantt_chart = []
sjf_ready_queue = []
sjf_index = 0
while sjf_ready_queue or sjf_index < numProcesses:
    if not sjf_ready_queue:
        heapq.heappush(sjf_ready_queue, (processes[sjf_index].cpu_burst, sjf_index))
        current_time = processes[sjf_index].arrival_time
        sjf_index += 1
    else:
        cpu_burst, index = heapq.heappop(sjf_ready_queue)
        process = processes[index]
        process.start_time = current_time
        current_time += process.cpu_burst + context_switch
        process.finish_time = current_time
        process.waiting_time = process.start_time - process.arrival_time
        process.turnaround_time = process.finish_time - process.arrival_time
        sjf_gantt_chart.append(GanttChart(process.pid, process.start_time, process.finish_time))
        while sjf_index < numProcesses and processes[sjf_index].arrival_time <= current_time:
            heapq.heappush(sjf_ready_queue, (processes[sjf_index].cpu_burst, sjf_index))
            sjf_index += 1

sjf_average_waiting_time = sum(process.waiting_time for process in processes) / numProcesses
sjf_average_turnaround_time = sum(process.turnaround_time for process in processes) / numProcesses
sjf_cpu_utilization = sum(process.cpu_burst for process in processes) / current_time
```

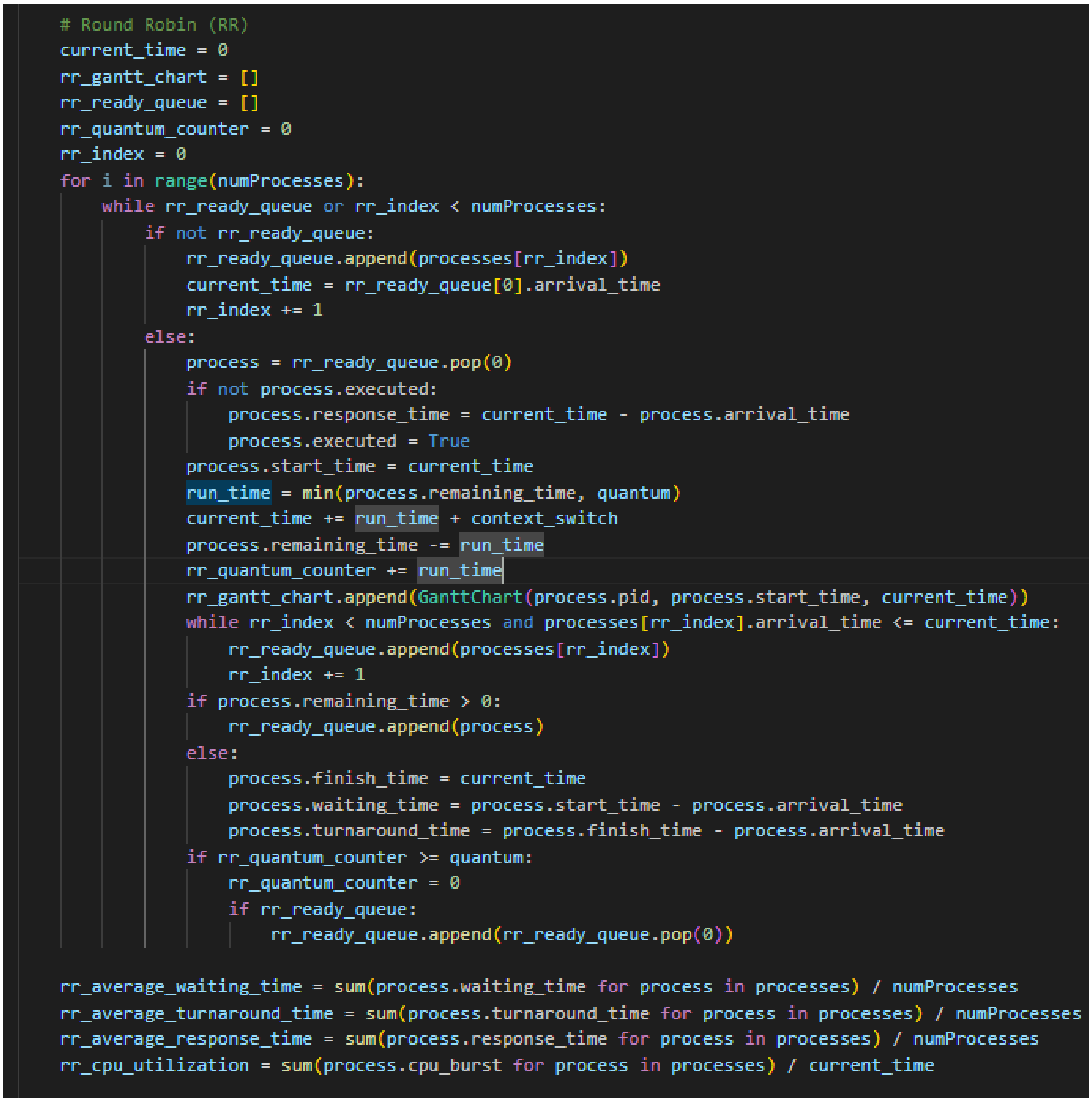
Code output :



```
Shortest Job First (SJF) (non-preemptive):
Gantt Chart:
PID: 1, Start Time: 0, End Time: 14
PID: 2, Start Time: 14, End Time: 23
PID: 3, Start Time: 23, End Time: 35
PID: 4, Start Time: 35, End Time: 42
PID: 5, Start Time: 42, End Time: 52
Finish Time     Waiting Time     Turnaround Time
20              15               20
65              50               53
62              40               43
44              11               15
74              36               40
Average Waiting Time = 4.0
Average Turnaround Time = 14.4
CPU Utilization = 0.8076923076923077
```

Simulates the scheduling of processes using Round-Robin (RR)

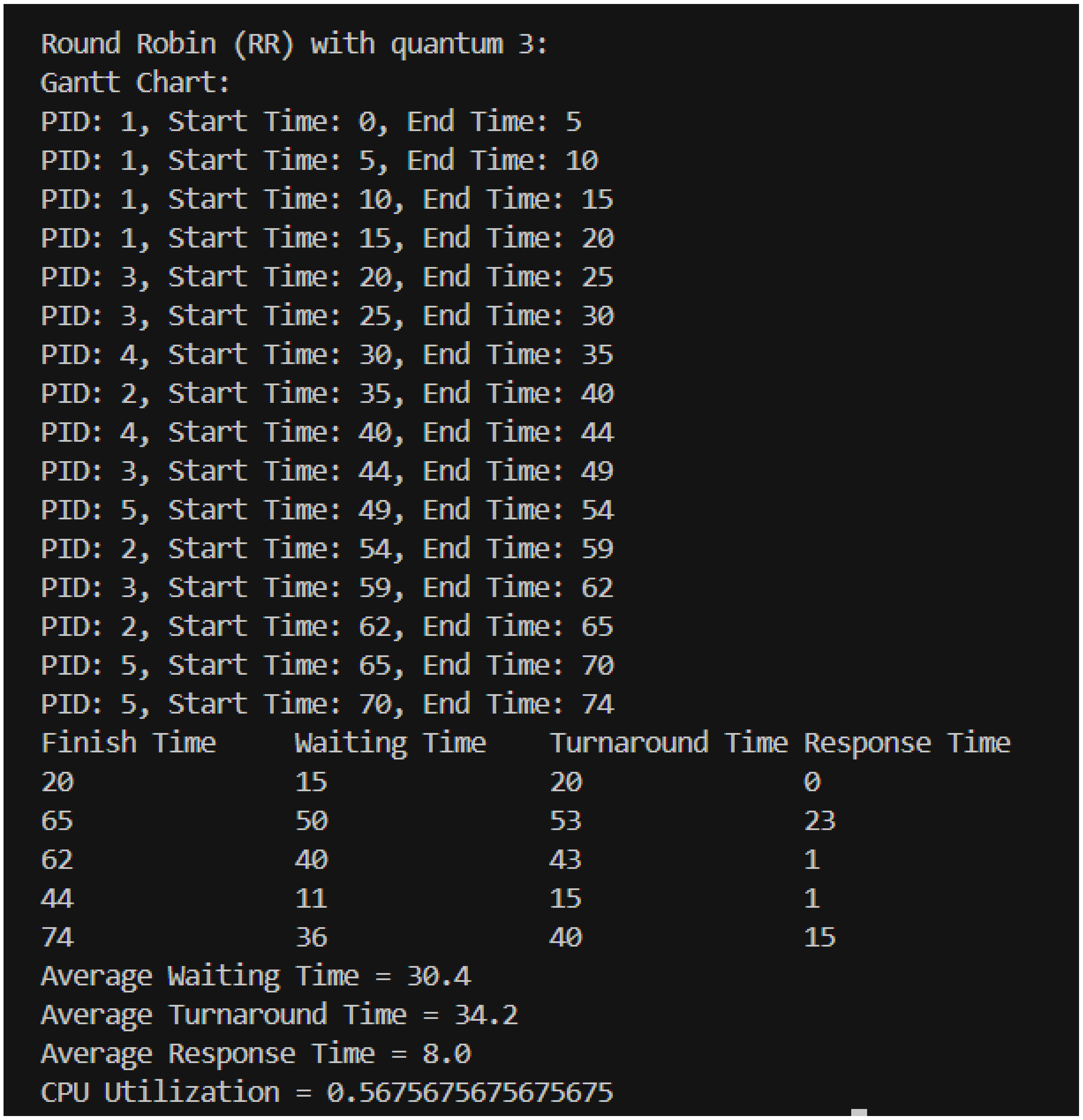
Python code :



```python
# Round Robin (RR)
current_time = 0
rr_gantt_chart = []
rr_ready_queue = []
rr_quantum_counter = 0
rr_index = 0
for i in range(numProcesses):
    while rr_ready_queue or rr_index < numProcesses:
        if not rr_ready_queue:
            rr_ready_queue.append(processes[rr_index])
            current_time = rr_ready_queue[0].arrival_time
            rr_index += 1
        else:
            process = rr_ready_queue.pop(0)
            if not process.executed:
                process.response_time = current_time - process.arrival_time
                process.executed = True
            process.start_time = current_time
            run_time = min(process.remaining_time, quantum)
            current_time += run_time + context_switch
            process.remaining_time -= run_time
            rr_quantum_counter += run_time
            rr_gantt_chart.append(GanttChart(process.pid, process.start_time, current_time))
            while rr_index < numProcesses and processes[rr_index].arrival_time <= current_time:
                rr_ready_queue.append(processes[rr_index])
                rr_index += 1
            if process.remaining_time > 0:
                rr_ready_queue.append(process)
            else:
                process.finish_time = current_time
                process.waiting_time = process.start_time - process.arrival_time
                process.turnaround_time = process.finish_time - process.arrival_time
            if rr_quantum_counter >= quantum:
                rr_quantum_counter = 0
                if rr_ready_queue:
                    rr_ready_queue.append(rr_ready_queue.pop(0))

rr_average_waiting_time = sum(process.waiting_time for process in processes) / numProcesses
rr_average_turnaround_time = sum(process.turnaround_time for process in processes) / numProcesses
rr_average_response_time = sum(process.response_time for process in processes) / numProcesses
rr_cpu_utilization = sum(process.cpu_burst for process in processes) / current_time
```

Code output :



```
Round Robin (RR) with quantum 3:
Gantt Chart:
PID: 1, Start Time: 0, End Time: 5
PID: 1, Start Time: 5, End Time: 10
PID: 1, Start Time: 10, End Time: 15
PID: 1, Start Time: 15, End Time: 20
PID: 3, Start Time: 20, End Time: 25
PID: 3, Start Time: 25, End Time: 30
PID: 4, Start Time: 30, End Time: 35
PID: 2, Start Time: 35, End Time: 40
PID: 4, Start Time: 40, End Time: 44
PID: 3, Start Time: 44, End Time: 49
PID: 5, Start Time: 49, End Time: 54
PID: 2, Start Time: 54, End Time: 59
PID: 3, Start Time: 59, End Time: 62
PID: 2, Start Time: 62, End Time: 65
PID: 5, Start Time: 65, End Time: 70
PID: 5, Start Time: 70, End Time: 74
Finish Time     Waiting Time    Turnaround Time Response Time
20              15              20              0
65              50              53              23
62              40              43              1
44              11              15              1
74              36              40              15
Average Waiting Time = 30.4
Average Turnaround Time = 34.2
Average Response Time = 8.0
CPU Utilization = 0.5675675675675675
```