Name: - Harsh Zanwar

Class: - AIDS-C / B1

Roll Number: - 77
PRN: - 12110333

Write a program to compute the finish time, turnaround time and waiting time for the following algorithms: a) list come list serve b) Shortest Job list (Preemptive and Non-Preemptive) c) Priority (Preemptive and Non-Preemptive) d) Round Robin

Code :-

```
class Process:
    def __init__(self, pid, arrival_time, burst_time, priority):
        self.pid = pid
        self.arrival_time = arrival_time
        self.burst_time = burst_time
        self.remaining_time = burst_time
        self.priority = priority

def fcfs(processes):
    processes.sort(key=lambda x: x.arrival_time)
    current_time = 0
    waiting_time = 0
    sequence = []
    for process in processes:
        waiting_time += max(0, current_time - process.arrival_time)
        current_time += process.burst_time
        sequence.append(process.pid)
    average_waiting_time = waiting_time / len(processes)
    print("Sequence of processes:", sequence)
    return average_waiting_time

def sjf(processes):
    processes.sort(key=lambda x: (x.arrival_time, x.burst_time))
    current_time = 0
    waiting_time = 0
```

```
    sequence = []
    for process in processes:
        waiting_time += max(0, current_time - process.arrival_time)
        current_time += process.burst_time
        sequence.append(process.pid)
    average_waiting_time = waiting_time / len(processes)
    print("Sequence of processes:", sequence)
    return average_waiting_time

def round_robin(processes, time_quantum):
    queue = []
    current_time = 0
    waiting_time = 0
    sequence = []
    remaining_processes = processes[:]
    while remaining_processes or queue:
        for process in remaining_processes[:]:
            if process.arrival_time <= current_time:
                queue.append(process)
                remaining_processes.remove(process)
        if queue:
            current_process = queue.pop(0)
            waiting_time += current_time - current_process.arrival_time
            if current_process.remaining_time <= time_quantum:
                current_time += current_process.remaining_time
                current_process.remaining_time = 0
            else:
                current_time += time_quantum
                current_process.remaining_time -= time_quantum
            queue.append(current_process)
            sequence.append(current_process.pid)
        else:
            current_time += 1
    average_waiting_time = waiting_time / len(processes)
    print("Sequence of processes:", sequence)
    return average_waiting_time

def priority_scheduling(processes, preemptive=False):
    processes.sort(key=lambda x: (x.arrival_time, x.priority))
    current_time = 0
    waiting_time = 0
    sequence = []
    remaining_processes = processes[:]
    while remaining_processes:
```

```
        ready_processes = [process for process in remaining_processes if
process.arrival_time <= current_time]
        if ready_processes:
            if preemptive:
                current_process = min(ready_processes, key=lambda x: x.priority)
                if current_process.remaining_time <= 0:
                    remaining_processes.remove(current_process)
                    waiting_time += max(0, current_time -
current_process.arrival_time)
                    current_time += current_process.burst_time
                    sequence.append(current_process.pid)
                else:
                    current_time += 1
                    current_process.remaining_time -= 1
                    sequence.append(current_process.pid)
            else:
                current_process = min(ready_processes, key=lambda x: x.priority)
                remaining_processes.remove(current_process)
                waiting_time += max(0, current_time -
current_process.arrival_time)
                current_time += current_process.burst_time
                sequence.append(current_process.pid)
        else:
            current_time += 1
    average_waiting_time = waiting_time / len(processes)
    print("Sequence of processes:", sequence)
    return average_waiting_time

def main():
    processes = []
    num_processes = int(input("Enter the number of processes: "))
    print("Total number of processes: ", num_processes)
    for i in range(num_processes):
        pid = i + 1
        arrival_time = int(input(f"Enter arrival time for process {pid}: "))
        burst_time = int(input(f"Enter burst time for process {pid}: "))
        processes.append(Process(pid, arrival_time, burst_time, None))
    while True:
        print("\nSelect a scheduling algorithm:")
        print("1. Shortest Job First (SJF)")
        print("2. First Come First Serve (FCFS)")
        print("3. Round Robin (RR)")
        print("4. Priority Scheduling (Preemptive)")
        print("5. Priority Scheduling (Non-preemptive)")
        print("6. Exit")
```

```
        choice = int(input("Enter your choice: "))
        if choice in range(1, 6):
            print("\nEntered input:")
            for process in processes:
                print(f"Process {process.pid}: Arrival Time =
{process.arrival_time}, Burst Time = {process.burst_time}")
            if choice in [4, 5]:
                for process in processes:
                    process.priority = int(input(f"Enter priority for process
{process.pid}: "))
            if choice == 1:
                print("\nAverage waiting time (SJF):", sjf(processes))
            elif choice == 2:
                print("\nAverage waiting time (FCFS):", fcfs(processes))
            elif choice == 3:
                time_quantum = int(input("Enter time quantum for Round Robin: "))
                print("\nAverage waiting time (RR):", round_robin(processes,
time_quantum))
            elif choice == 4:
                print("\nAverage waiting time (Priority, Preemptive):",
priority_scheduling(processes, True))
            elif choice == 5:
                print("\nAverage waiting time (Priority, Non-preemptive):",
priority_scheduling(processes, False))
            elif choice == 6:
                break
            else:
                print("\nInvalid choice. Please try again.")

if __name__== "__main__":
    main()
```

Output :-

```
PS C:\TY\Operating System> & C:/Users/yoges/anaconda3/python.exe "c:/TY/Operating System/Scheduling.py"
Enter the number of processes: 3
Total number of processes:  3
Enter arrival time for process 1: 0
Enter burst time for process 1: 5
Enter arrival time for process 2: 1
Enter burst time for process 2: 3
Enter arrival time for process 3: 0
Enter burst time for process 3: 4

Select a scheduling algorithm:
1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Round Robin (RR)
4. Priority Scheduling (Preemptive)
5. Priority Scheduling (Non-preemptive)
6. Exit
Enter your choice: 1

Entered input:
Process 1: Arrival Time = 0, Burst Time = 5
Process 2: Arrival Time = 1, Burst Time = 3
Process 3: Arrival Time = 0, Burst Time = 4
Sequence of processes: [3, 1, 2]

Average waiting time (SJF): 4.0

Select a scheduling algorithm:
1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Round Robin (RR)
4. Priority Scheduling (Preemptive)
5. Priority Scheduling (Non-preemptive)
6. Exit
Enter your choice: 2

Entered input:
Process 3: Arrival Time = 0, Burst Time = 4
Process 1: Arrival Time = 0, Burst Time = 5
Process 2: Arrival Time = 1, Burst Time = 3
Sequence of processes: [3, 1, 2]

Average waiting time (FCFS): 4.0
```

```
Select a scheduling algorithm:
1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Round Robin (RR)
4. Priority Scheduling (Preemptive)
5. Priority Scheduling (Non-preemptive)
6. Exit
Enter your choice: 3

Entered input:
Process 3: Arrival Time = 0, Burst Time = 4
Process 1: Arrival Time = 0, Burst Time = 5
Process 2: Arrival Time = 1, Burst Time = 3
Enter time quantum for Round Robin: 2
Sequence of processes: [3, 1, 3, 2, 1, 2, 1]

Average waiting time (RR): 13.0

Select a scheduling algorithm:
1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Round Robin (RR)
4. Priority Scheduling (Preemptive)
5. Priority Scheduling (Non-preemptive)
6. Exit
Enter your choice: 4

Entered input:
Process 3: Arrival Time = 0, Burst Time = 4
Enter priority for process 3: 3
Process 1: Arrival Time = 0, Burst Time = 5
Enter priority for process 1: 2
Process 2: Arrival Time = 1, Burst Time = 3
Enter priority for process 2: 1
Sequence of processes: [1, 2, 3]

Average waiting time (Priority, Preemptive): 4.0
```

```
Select a scheduling algorithm:
1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Round Robin (RR)
4. Priority Scheduling (Preemptive)
5. Priority Scheduling (Non-preemptive)
6. Exit
Enter your choice: 3

Entered input:
Process 3: Arrival Time = 0, Burst Time = 4
Process 1: Arrival Time = 0, Burst Time = 5
Process 2: Arrival Time = 1, Burst Time = 3
Enter time quantum for Round Robin: 2
Sequence of processes: [3, 1, 3, 2, 1, 2, 1]

Average waiting time (RR): 13.0

Select a scheduling algorithm:
1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Round Robin (RR)
```