Operating Systems - Assignment 3

Aman Aggarwal 102165010 3NC5

Code:-

```
def calculate_waiting_time(processes, n, burst_time, waiting_time):
    waiting_time[0] = 0
    for i in range(1, n):
        waiting_time[i] = burst_time[i - 1] + waiting_time[i - 1]

def find_waiting_time_sjf(processes, n, burst_time, waiting_time):
    remaining_time = [0] * n
    for i in range(n):
        remaining_time[i] = burst_time[i]

    complete = 0
    t = 0
    minm = float('inf')
    shortest = 0
    check = False

    while complete != n:
        for j in range(n):
            if (processes[j][1] <= t and remaining_time[j] < minm and
remaining_time[j] > 0):
                minm = remaining_time[j]
                shortest = j
                check = True

        if not check:
            t += 1
            continue

        remaining_time[shortest] -= 1

        if remaining_time[shortest] == 0:
            complete += 1
            minm = float('inf')

        for j in range(n):
            if (processes[j][1] <= t and remaining_time[j] != 0):
                waiting_time[j] += 1

        t += 1

def find_waiting_time_priority(processes, n, burst_time, waiting_time,
priority):
    burst_time.sort()
    for i in range(n):
        for j in range(n):
            if processes[j][0] == burst_time[i]:
                index = j
                break

        waiting_time[index] = 0
```

```
        for j in range(i):
            waiting_time[index] += burst_time[j]

def find_waiting_time_rr(processes, n, burst_time, waiting_time,
quantum):
    remaining_time = [0] * n
    for i in range(n):
        remaining_time[i] = burst_time[i]

    t = 0
    while True:
        done = True
        for i in range(n):
            if remaining_time[i] > 0:
                done = False
                if remaining_time[i] > quantum:
                    t += quantum
                    remaining_time[i] -= quantum
                else:
                    t += remaining_time[i]
                    waiting_time[i] = t - burst_time[i]
                    remaining_time[i] = 0

        if done:
            break

def find_waiting_time_fcfs(processes, n, burst_time, waiting_time):
    waiting_time[0] = 0
    for i in range(1, n):
        waiting_time[i] = burst_time[i - 1] + waiting_time[i - 1]

def find_average_waiting_time(processes, n, burst_time, waiting_time):
    total_waiting_time = sum(waiting_time)
    return total_waiting_time / n

def execute_scheduling_algorithm(processes, n, burst_time, arrival_time,
priority, quantum):
    waiting_time = [0] * n

    if quantum > 0:
        find_waiting_time_rr(processes, n, burst_time, waiting_time,
quantum)
    elif priority:
        find_waiting_time_priority(processes, n, burst_time,
waiting_time, priority)
    elif arrival_time:
        find_waiting_time_sjf(processes, n, burst_time, waiting_time)
    else:
        find_waiting_time_fcfs(processes, n, burst_time, waiting_time)

    avg_waiting_time = find_average_waiting_time(processes, n,
burst_time, waiting_time)

    print("Process\tBurst Time\tWaiting Time")
    for i in range(n):
        print(f"P{i + 1}\t\t{burst_time[i]}\t\t{waiting_time[i]}")

    print(f"Average Waiting Time: {avg_waiting_time:.2f}")
```

```
def main():
    n = int(input("Enter the number of processes: "))

    processes = []
    burst_time = []
    arrival_time = False
    priority = None
    quantum = 0

    for i in range(n):
        burst = int(input(f"Enter burst time for Process P{i + 1}: "))
        burst_time.append(burst)

        arrival = input(f"Is there an arrival time for Process P{i + 1}?
(y/n): ").lower()
        if arrival == 'y':
            arrival_time = True
            arrival_val = int(input(f"Enter arrival time for Process P{i
+ 1}: "))
        else:
            arrival_val = 0

        if arrival_time:
            processes.append((burst, arrival_val))
        else:
            processes.append((burst, 0))

        if not priority:
            priority = input(f"Is there a priority for Process P{i + 1}?
(y/n): ").lower()
            if priority == 'y':
                priority_val = int(input(f"Enter priority for Process P{i
+ 1}: "))
            else:
                priority_val = None
        if priority:
            if priority_val is not None:
                processes[i] = (processes[i][0], processes[i][1],
priority_val)

    algorithm = input("Select the scheduling algorithm:\n"
                      "a) Round Robin Scheduling (RRS)\n"
                      "b) Shortest job first Scheduling (SJFS)\n"
                      "c) First Come First Serve Scheduling (FCFS)\n"
                      "d) Priority Scheduling (PS)\n"
                      "Enter the letter (a/b/c/d): ").lower()

    if algorithm == 'a':
        quantum = int(input("Enter time quantum for Round Robin
Scheduling: "))
    elif algorithm == 'b':
        arrival_time = True
    elif algorithm == 'd':
        priority = True

    processes.sort(key=lambda x: x[1])
```

```
    if algorithm == 'b' or algorithm == 'd':
        processes.sort(key=lambda x: x[0])

    print("\nExecuting Scheduling Algorithm...\n")
    execute_scheduling_algorithm(processes, n, burst_time, arrival_time,
priority, quantum)

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

```
Enter the number of processes: 5
Enter burst time for Process P1: 2
Is there an arrival time for Process P1? (y/n): y
Enter arrival time for Process P1: 0
Is there a priority for Process P1? (y/n): n
Enter burst time for Process P2: 5
Is there an arrival time for Process P2? (y/n): y
Enter arrival time for Process P2: 2
Enter burst time for Process P3: 6
Is there an arrival time for Process P3? (y/n): y
Enter arrival time for Process P3: 3
Enter burst time for Process P4: 4
Is there an arrival time for Process P4? (y/n): y
Enter arrival time for Process P4: 2
Enter burst time for Process P5: 6
Is there an arrival time for Process P5? (y/n): y
Enter arrival time for Process P5: 8
Select the scheduling algorithm:
a) Round Robin Scheduling (RRS)
b) Shortest job first Scheduling (SJFS)
c) First Come First Serve Scheduling (FCFS)
d) Priority Scheduling (PS)
Enter the letter (a/b/c/d): c

Executing Scheduling Algorithm...

Process Burst Time      Waiting Time
P1              2               0
P2              4               6
P3              5               2
P4              6               17
P5              6               0
Average Waiting Time: 5.00
```