

**Experiment-6**

Operating Systems

**Program:**

MCA

**SVKM'S-NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**School of Technology Management and Engineering**

**[2024-25]**

Lab Manual

PART B

| Roll No.: A073 | Name: Aryan Srivastava |
| --- | --- |
| Class: MCA | Batch: B3 |
| Experiment Number | |
| Date of Experiment: | Date of Submission: |
| Grade: |  |

**B.1 Program with Output to be written by student**

**CODE**

def round\_robin\_scheduling(processes, burst\_times, arrival\_times, time\_quantum):

    n = len(processes)

    remaining\_times = burst\_times[:]

    time = 0

    completed\_processes = []

    process\_queue = []

    gantt\_chart = []

    completion\_times = [0] \* n

    for i in range(n):

        if arrival\_times[i] <= time:

            process\_queue.append(i)

    while process\_queue:

        process\_index = process\_queue.pop(0)

        if remaining\_times[process\_index] <= time\_quantum:

            gantt\_chart.append((processes[process\_index], time, time + remaining\_times[process\_index]))

            time += remaining\_times[process\_index]

            remaining\_times[process\_index] = 0

            completion\_times[process\_index] = time

            completed\_processes.append((processes[process\_index], time))

        else:

            gantt\_chart.append((processes[process\_index], time, time + time\_quantum))

            time += time\_quantum

            remaining\_times[process\_index] -= time\_quantum

            process\_queue.append(process\_index)

        for i in range(n):

            if arrival\_times[i] <= time and remaining\_times[i] > 0 and i not in process\_queue:

                process\_queue.append(i)

    waiting\_times = [0] \* n

    turnaround\_times = [0] \* n

    for i in range(n):

        turnaround\_times[i] = completion\_times[i] - arrival\_times[i]

        waiting\_times[i] = turnaround\_times[i] - burst\_times[i]

    average\_waiting\_time = sum(waiting\_times) / n

    average\_turnaround\_time = sum(turnaround\_times) / n

    return completed\_processes, gantt\_chart, waiting\_times, turnaround\_times, average\_waiting\_time, average\_turnaround\_time

def draw\_gantt\_chart(gantt\_chart):

    chart = ""

    for process, start\_time, end\_time in gantt\_chart:

        chart += f"{start\_time} --> {process} " + "-" \* (end\_time - start\_time) + f" {end\_time}\n"

    return chart

number\_of\_processes = 5

# number\_of\_processes = int(input("Enter the number of processes: "))

processes = ['p1','p2','p3','p4','p5']

burst\_times = [5, 3, 1, 2, 3]

arrival\_times = [0, 1, 2, 3, 4]

# for i in range(number\_of\_processes):

#     process\_name = input(f"Enter the name of process {i + 1}: ")

#     burst\_time = int(input(f"Enter the burst time for {process\_name}: "))

#     arrival\_time = int(input(f"Enter the arrival time for {process\_name}: "))

#     processes.append(process\_name)

#     burst\_times.append(burst\_time)

#     arrival\_times.append(arrival\_time)

time\_quantum = int(input("Enter the time quantum: "))

completed\_processes, gantt\_chart, waiting\_times, turnaround\_times, average\_waiting\_time, average\_turnaround\_time = round\_robin\_scheduling(

    processes, burst\_times, arrival\_times, time\_quantum

)

print("Completed Processes:", completed\_processes)

gantt\_chart\_text = draw\_gantt\_chart(gantt\_chart)

print("\nGantt Chart:")

print(gantt\_chart\_text)

print("Waiting Times:", waiting\_times)

print("Turnaround Times:", turnaround\_times)

print(f"Average Waiting Time: {average\_waiting\_time:.2f}")

print(f"Average Turnaround Time: {average\_turnaround\_time:.2f}")

**OUTPUT**

