DIVYANSHU MAKHARIA
221070036

ASSIGNMENT 2

AIM:CPU scheduling algorithms.

THEORY:

CPU scheduling is a crucial aspect of operating system design, as it determines how tasks or processes are assigned to the CPU for execution. Here's a brief overview and some theoretical considerations for each of the mentioned CPU scheduling algorithms:

First-Come, First-Served (FCFS):

- FCFS is the simplest scheduling algorithm, where processes are executed in the order they arrive in the ready queue.
- It's typically not suitable for time-sharing systems or environments where responsiveness is critical.

Shortest Job First (SJF):

- SJF schedules processes based on their burst time, executing the shortest job first.
- This algorithm minimizes average waiting time and turnaround time, leading to efficient resource utilization.
- However, it's challenging to predict the burst time accurately, especially in a real-time or interactive system.

Round Robin (RR):

- RR is a preemptive scheduling algorithm where each process is assigned a fixed time slice (quantum) for execution.
- Processes are executed in a circular queue, and if a process doesn't complete within its time slice, it's moved to the back of the queue.
- RR provides fair scheduling and prevents starvation since every process gets CPU time.

Priority Scheduling:

- Priority scheduling assigns priorities to processes, and the CPU is allocated to the process with the highest priority.
- It can be either preemptive or non-preemptive.

DIVYANSHU MAKHARIA
221070036

CODE:

```
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
def fcfs(processes):
    processes.sort(key=lambda x: x[0])
    n = len(processes)
    timeline = []
    current_time = 0
    for process in processes:
        timeline.append((process[0], current_time,
process[1],process[3]))
        current_time += process[1]

    waiting_time = [0] * n
    turnaround_time =[0]*n
    waiting_time[0] = 0
    turnaround_time[0] = processes[0][1]

    for i in range(1, n):
        waiting_time[i] = turnaround_time[i - 1]
        turnaround_time[i] = waiting_time[i] + processes[i][1]


    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n


    print("FCFS Scheduling")
    print("Process\t\tArrival Time\t\tBurst Time\t\tWaiting
Time\t\tTurnaround Time")
    for i in range(n):

print(f"{processes[i][3]}\t\t{processes[i][0]}\t\t\t{processes[i][1]}\
t\t\t{waiting_time[i]}\t\t\t{turnaround_time[i]}")

    print(f"\nAverage Waiting Time: {avg_waiting_time}")
    print(f"Average Turnaround Time: {avg_turnaround_time}")
    return timeline
```

```
def sjf(processes):
    processes.sort(key=lambda x: (x[0],x[1]))
    n = len(processes)
    timeline = []

    current_time = 0
    for process in processes:
        timeline.append((current_time, current_time + process[1],
process[1],process[3]))
        current_time += process[1]

    waiting_time = [0] * n
    turnaround_time = [0] * n

    waiting_time[0] = 0
    turnaround_time[0] = processes[0][1]

    for i in range(1, n):
        waiting_time[i] = turnaround_time[i - 1]
        turnaround_time[i] = waiting_time[i] + processes[i][1]

    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n

    print("SJF Scheduling:")
    print("Process\t\tBurst Time\t\tWaiting Time\t\tTurnaround Time")
    for i in range(n):

print(f"{processes[i][3]}\t\t{processes[i][1]}\t\t\t{waiting_time[i]}\
t\t\t{turnaround_time[i]}")

    print(f"\nAverage Waiting Time: {avg_waiting_time}")
    print(f"Average Turnaround Time: {avg_turnaround_time}\n")

    return timeline

def round_robin(processes):
```

```
    time_quantum = 2
    n = len(processes)
    rem_time = [process[1] for process in processes]
    process_name=[process[3] for process in processes]
    waiting_time = [0] * n
    turnaround_time = [0] * n
    total_executed_time = 0
    current_time = 0
    timeline=[]

    while any(rem_time):
        for i in range(n):
            if rem_time[i] > 0:
                execution_time = min(rem_time[i], time_quantum)
                timeline.append((i + 1, current_time,
execution_time,process_name[i]))
                current_time += execution_time
                rem_time[i] -= execution_time
    return timeline


def priority_scheduling(processes):
    processes.sort(key=lambda x: (x[0],x[2]))
    n = len(processes)
    waiting_time = [0] * n
    turnaround_time = [0] * n

    waiting_time[0] = 0
    turnaround_time[0] = processes[0][1]

    for i in range(1, n):
        waiting_time[i] = turnaround_time[i - 1]
        turnaround_time[i] = waiting_time[i] + processes[i][1]

    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n

    print("Priority Scheduling:")
```

```
    print("Process\t\tPriority\t\tBurst Time\t\tWaiting
Time\t\tTurnaround Time")
    for i in range(n):

print(f"{processes[i][3]}\t\t{processes[i][2]}\t\t\t{processes[i][1]}\
t\t\t{waiting_time[i]}\t\t\t{turnaround_time[i]}")

    print(f"\nAverage Waiting Time: {avg_waiting_time}")
    print(f"Average Turnaround Time: {avg_turnaround_time}\n")
    return timeline

def plot_gantt_chart(timeline,title):
    fig, ax = plt.subplots(figsize=(10, 1))

    for i,entry in enumerate(timeline):
        rect = Rectangle((entry[1], 0), entry[2], 1,
edgecolor='black', facecolor=f'C{i}', label=f'{entry[3]}')
        ax.add_patch(rect)

    ax.set_xlim(0, max(entry[1] + entry[2] for entry in timeline) + 2)
    ax.set_ylim(0, 1)
    ax.set_yticks([])

    plt.title(title)
    plt.xlabel('Time')
    plt.legend(loc='upper right', bbox_to_anchor=(1.12, 1))
    plt.show()

if __name__ == "__main__":
    processes=[]
    pro=int(input("Enter number of processes-"))
    for x in range(1,pro+1):
        process_name=input("Enter process name-")
        arr_time=int(input("Arrrival Time for the process-"))
        burst_time=int(input("Execution time for the process in
sec-"))
        priority=int(input("Priority of the process(Between 1 and "+
str(pro) + ")"))
```

```
        processes.append((arr_time,burst_time,priority,process_name))
    while True:
        model=int(input("Which algorithm do you wanna
use\n1-FCFS\n2-SJF\n3-RR\n4-Priority\n5-Exit\nEnter Option-"))
        if(model==1):
            timeline=fcfs(processes)
            plot_gantt_chart(timeline,'FCFS GANTT CHART')
        elif(model==2):
            timeline=sjf(processes)
            plot_gantt_chart(timeline,'SJF GANTT CHART')
        elif(model==3):
            timeline=round_robin(processes)
            plot_gantt_chart(timeline,'Round Robin GANTT CHART')
        elif(model==4):
            timeline=priority_scheduling(processes)
            plot_gantt_chart(timeline,'PRIORITY GANTT CHART')
        elif(model==5):
         break
        else:
            print("No other method available")
    print('Exit')
```

DIVYANSHU MAKHARIA
221070036

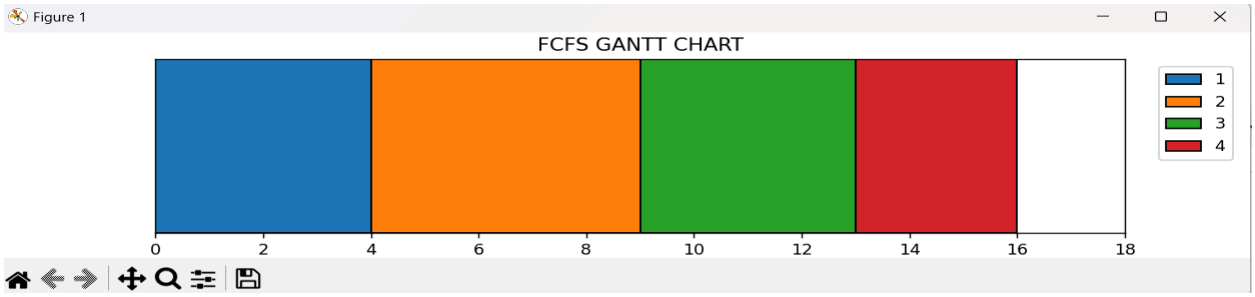
OUTPUT:

```
PS D:\OS_lab> & C:/Users/Divyanshu/AppData/Local/Microsoft/WindowsApp
 Enter number of processes-4
 Enter process name-1
 Arrrival Time for the process-0
 Execution time for the process in sec-4
 Priority of the process(Between 1 and 4)2
 Enter process name-2
 Arrrival Time for the process-2
 Execution time for the process in sec-5
 Priority of the process(Between 1 and 4)2
 Enter process name-3
 Arrrival Time for the process-2
 Execution time for the process in sec-4
 Priority of the process(Between 1 and 4)1
 Enter process name-4
 Arrrival Time for the process-6
 Execution time for the process in sec-3
 Priority of the process(Between 1 and 4)4
 Which algorithm do you wanna use
 1-FCFS
 2-SJF
 3-RR
 4-Priority
 5-Exit
 Enter Option-1
 FCFS Scheduling
```

FIRST COME FIRST SERVE:

```
Process         Arrival Time            Burst Time              Waiting Time            Turnaround Time
1               0                       4                       0                       4
2               2                       5                       4                       9
3               2                       4                       9                       13
4               6                       3                       13                      16

Average Waiting Time: 6.5
Average Turnaround Time: 10.5
```

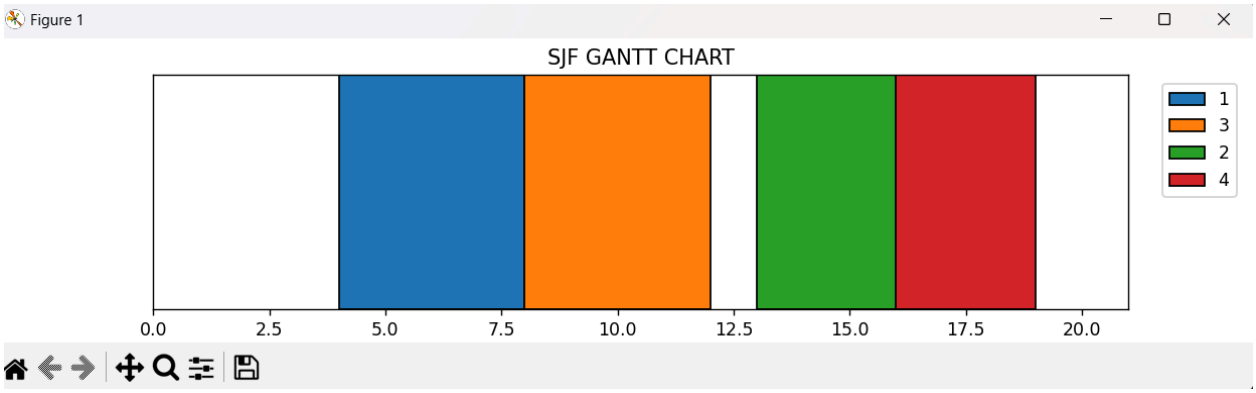


DIVYANSHU MAKHARIA
221070036

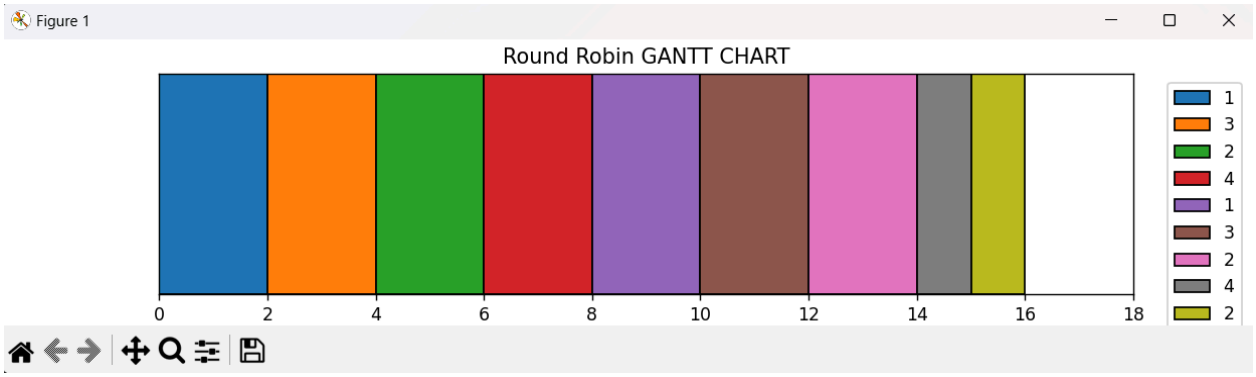
SHORTEST JOB FIRST:

```
Which algorithm do you wanna use
1-FCFS
2-SJF
3-RR
4-Priority
5-Exit
Enter Option-2
SJF Scheduling:
Process         Burst Time              Waiting Time            Turnaround Time
1               4                       0                       4
3               4                       4                       8
2               5                       8                       13
4               3                       13                      16

Average Waiting Time: 6.25
Average Turnaround Time: 10.25
```



ROUND ROBIN:

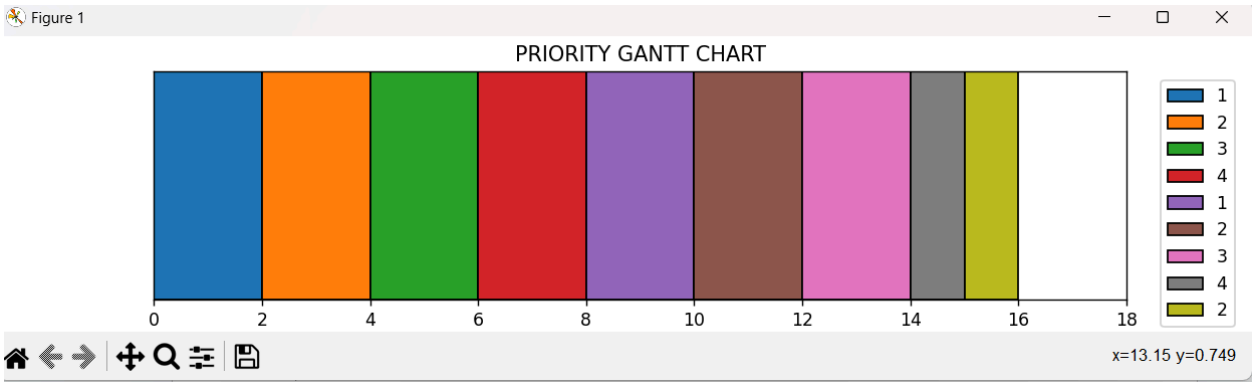


DIVYANSHU MAKHARIA
221070036

PRIORITY SCHEDULING:

```
Which algorithm do you wanna use
1-FCFS
2-SJF
3-RR
4-Priority
5-Exit
Enter Option-4
Priority Scheduling:
Process         Priority              Burst Time            Waiting Time          Turnaround Time
1               2                     4                     0                     4
3               1                     4                     4                     8
2               2                     5                     8                     13
4               4                     3                     13                    16

Average Waiting Time: 6.25
Average Turnaround Time: 10.25
```



CONCLUSION: Thus,we have learnt about CPU Scheduling algorithms and implemented them.