Assignment 4

Operating Systems

Submitted to:

Department of Computer Science Engineering

Punjab Engineering College (Deemed to be University)

Chandigarh

Submitted by :

Navneet Yadav

SID: 21105127

Branch: ECE

Assignment 4

Navneet Yadav (21105127)

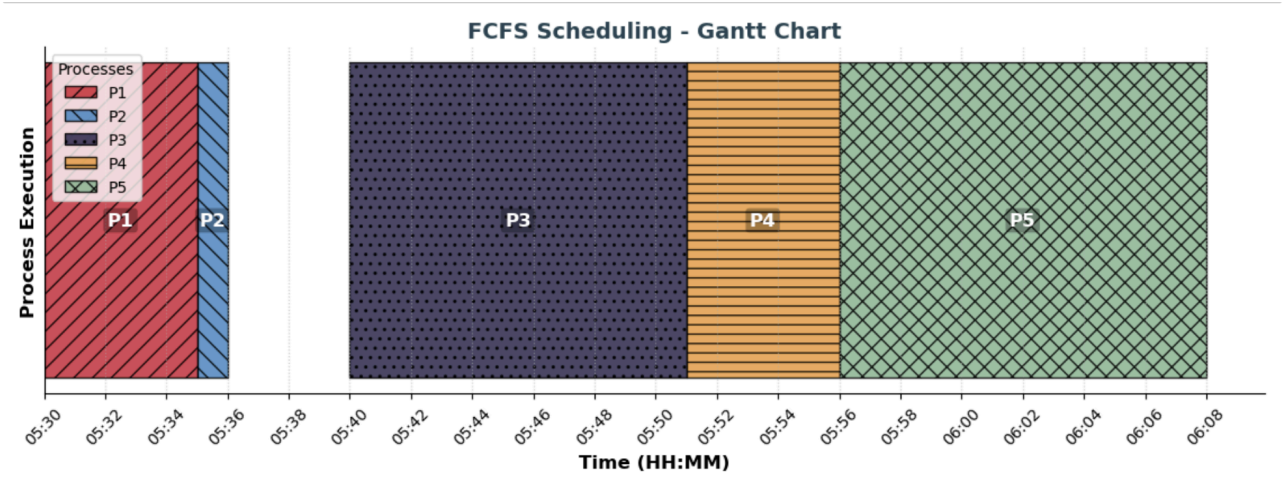
The arrival time for four processes to a system are as below

Process ID	Arrival Time	Estimated Processing time
P1	5:30 PM	5 unit time
P2	5:33 PM	1 unit time
P3	5:40 PM	11 unit time
P4	5:42 PM	5 unit time
P5	5:45 PM	12 unit time

1. Multiprogramming System

```
import matplotlib.pyplot as plt
import numpy as np

processes = [
    ("P1", 5 * 60 + 30, 5),
    ("P2", 5 * 60 + 33, 1),
    ("P3", 5 * 60 + 40, 11),
    ("P4", 5 * 60 + 42, 5),
    ("P5", 5 * 60 + 45, 12)
]

fcfs_schedule = sorted(processes, key=lambda x: x[1])

start_time = fcfs_schedule[0][1]
gantt_fcfs = []
for process in fcfs_schedule:
    process_id, arrival, burst = process
    if start_time < arrival:
        start_time = arrival
    end_time = start_time + burst
    gantt_fcfs.append((process_id, start_time, end_time))
    start_time = end_time

def minutes_to_time(minutes):
```

```
    hours = minutes // 60
    mins = minutes % 60
    return f"{hours:02d}:{mins:02d}"

time_range = range(gantt_fcfs[0][1], gantt_fcfs[-1][2] + 1, 1)
time_labels = [minutes_to_time(t) for t in time_range]

colors = ["#D72638", "#3F88C5", "#2E294E", "#F49D37", "#83B692"]
hatch_patterns = ['//', '\\\\', '..', '--', 'xx']

plt.figure(figsize=(14, 4))

for i, (process_id, start, end) in enumerate(gantt_fcfs):
    plt.barh(y=0, width=end - start, left=start, height=0.5,
             color=colors[i % len(colors)], edgecolor="black", alpha=0.85,
             hatch=hatch_patterns[i % len(hatch_patterns)], label=process_id)

    plt.text((start + end) / 2, 0, process_id, ha="center", va="center",
             fontsize=12, color="white", fontweight="bold", bbox=dict(facecolor='black', alpha=0.3,
edgecolor='none', boxstyle="round,pad=0.2"))

plt.xticks(time_range[::2], time_labels[::2], rotation=45, fontsize=10)
plt.xlabel("Time (HH:MM)", fontsize=12, fontweight="bold")
plt.ylabel("Process Execution", fontsize=12, fontweight="bold")
plt.title("FCFS Scheduling - Gantt Chart", fontsize=14, fontweight="bold", color="#264653")

plt.yticks([])
plt.legend(loc="upper left", title="Processes", frameon=True, fontsize=10)
plt.grid(axis='x', linestyle=":", alpha=0.6, linewidth=0.8)
plt.gca().spines["top"].set_visible(False)
plt.gca().spines["right"].set_visible(False)

plt.show()
```

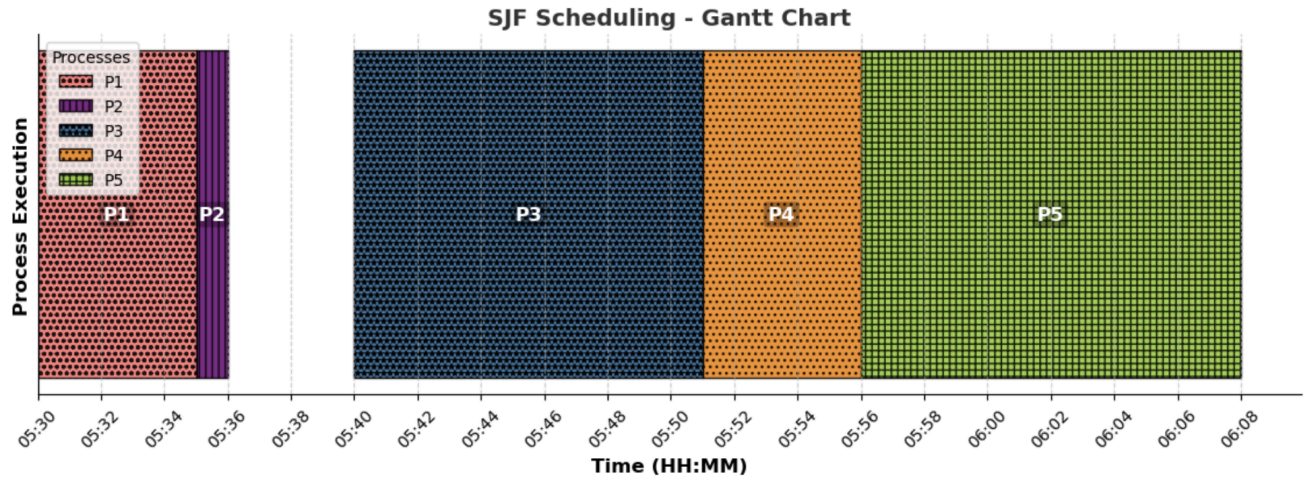


2. Multiprogramming with Shortest Job First (Non-Preemptive)

```
import matplotlib.pyplot as plt
import numpy as np

processes = [
    ("P1", 5 * 60 + 30, 5),
    ("P2", 5 * 60 + 33, 1),
    ("P3", 5 * 60 + 40, 11),
    ("P4", 5 * 60 + 42, 5),
    ("P5", 5 * 60 + 45, 12)
]

sjf_schedule = sorted(processes, key=lambda x: x[2])

current_time = 0
gantt_sjf = []
completed_processes = []

while len(completed_processes) < len(processes):
    eligible_processes = [p for p in sjf_schedule if p[1] <= current_time and p not in
completed_processes]

    if not eligible_processes:
        current_time += 1
```

```
        continue

    next_process = min(eligible_processes, key=lambda x: x[2])

    process_id, arrival, burst = next_process
    start_time = max(current_time, arrival)
    end_time = start_time + burst
    gantt_sjf.append((process_id, start_time, end_time))
    completed_processes.append(next_process)
    current_time = end_time

def minutes_to_time(minutes):
    hours = minutes // 60
    mins = minutes % 60
    return f"{hours:02d}:{mins:02d}"

time_range = range(gantt_sjf[0][1], gantt_sjf[-1][2] + 1, 1)
time_labels = [minutes_to_time(t) for t in time_range]

colors = ["#FF6B6B", "#6A0572", "#1E6091", "#F77F00", "#8AC926"]
hatch_patterns = ['ooo', '|||', '***', '...', '+++']

plt.figure(figsize=(14, 4))

for i, (process_id, start, end) in enumerate(gantt_sjf):
    plt.barh(y=0, width=end - start, left=start, height=0.5,
             color=colors[i % len(colors)], edgecolor="black", alpha=0.85,
             hatch=hatch_patterns[i % len(hatch_patterns)], label=process_id)

    plt.text((start + end) / 2, 0, process_id, ha="center", va="center",
             fontsize=12, color="white", fontweight="bold",
             bbox=dict(facecolor='black', alpha=0.3, edgecolor='none', boxstyle="round,pad=0.2"))

plt.xticks(time_range[::2], time_labels[::2], rotation=45, fontsize=10)
plt.xlabel("Time (HH:MM)", fontsize=12, fontweight="bold")
plt.ylabel("Process Execution", fontsize=12, fontweight="bold")
plt.title("SJF Scheduling - Gantt Chart", fontsize=14, fontweight="bold", color="#333533")

plt.yticks([])
plt.legend(loc="upper left", title="Processes", frameon=True, fontsize=10)
```

```
plt.grid(axis='x', linestyle="--", alpha=0.6, linewidth=0.8)
plt.gca().spines["top"].set_visible(False)
plt.gca().spines["right"].set_visible(False)

plt.show()
```

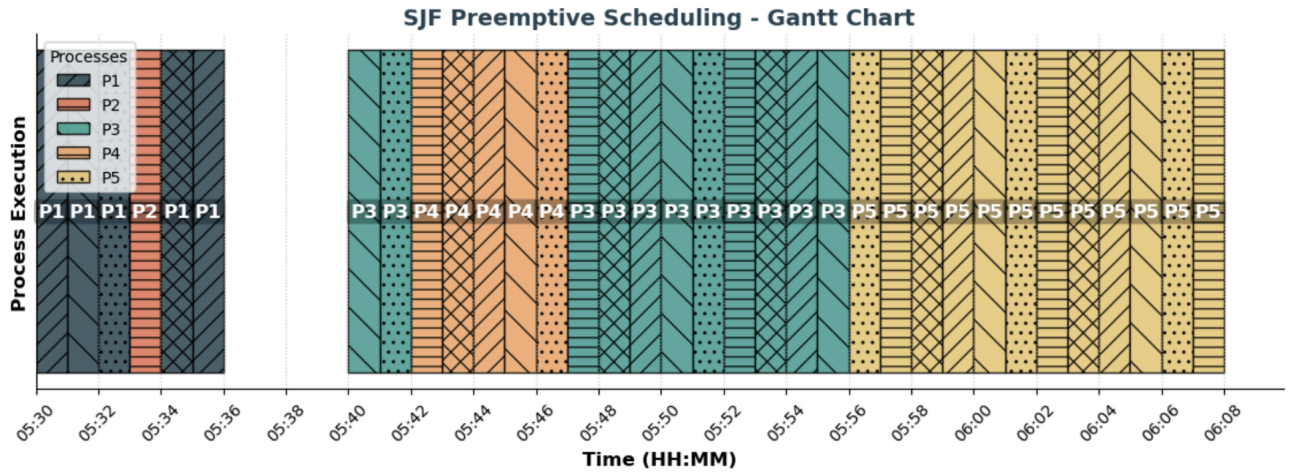


3. Multiprogramming with Shortest Job First (Preemptive)

```
import matplotlib.pyplot as plt

processes = [
    ("P1", 5 * 60 + 30, 5),
    ("P2", 5 * 60 + 33, 1),
    ("P3", 5 * 60 + 40, 11),
    ("P4", 5 * 60 + 42, 5),
    ("P5", 5 * 60 + 45, 12)
]

colors = {
    "P1": "#264653",
    "P2": "#E76F51",
    "P3": "#2A9D8F",
    "P4": "#F4A261",
    "P5": "#E9C46A"
}

hatch_patterns = ['//', '\\', '..', '--', 'xx']
```

```
remaining_time = {p[0]: p[2] for p in processes}
completed_processes = []
current_time = min(p[1] for p in processes)
gantt_sjf_preemptive = []

while len(completed_processes) < len(processes):
    available_processes = [p for p in processes if p[1] <= current_time and p[0] not in
completed_processes]
    if available_processes:
        next_process = min(available_processes, key=lambda p: remaining_time[p[0]])
        process_id = next_process[0]
        gantt_sjf_preemptive.append((process_id, current_time, current_time + 1))
        remaining_time[process_id] -= 1
        current_time += 1
        if remaining_time[process_id] == 0:
            completed_processes.append(process_id)
    else:
        current_time += 1

def minutes_to_time(minutes):
    hours = minutes // 60
    mins = minutes % 60
    return f"{hours:02d}:{mins:02d}"

time_range = range(gantt_sjf_preemptive[0][1], gantt_sjf_preemptive[-1][2] + 1, 1)
time_labels = [minutes_to_time(t) for t in time_range]

plt.figure(figsize=(14, 4))

for i, (process_id, start, end) in enumerate(gantt_sjf_preemptive):
    plt.barh(y=0, width=end-start, left=start, height=0.5,
             color=colors[process_id], edgecolor="black", alpha=0.85,
             hatch=hatch_patterns[i % len(hatch_patterns)], label=process_id if process_id not in
[p[0] for p in gantt_sjf_preemptive[:i]] else "")
    plt.text((start + end) / 2, 0, process_id, ha="center", va="center",
             fontsize=12, color="white", fontweight="bold",
             bbox=dict(facecolor='black', alpha=0.3, edgecolor='none', boxstyle="round,pad=0.2"))

plt.xticks(time_range[::2], time_labels[::2], rotation=45, fontsize=10)
```

```
plt.xlabel("Time (HH:MM)", fontsize=12, fontweight="bold")
plt.ylabel("Process Execution", fontsize=12, fontweight="bold")
plt.title("SJF Preemptive Scheduling - Gantt Chart", fontsize=14, fontweight="bold",
color="#264653")
plt.yticks([])
plt.legend(loc="upper left", title="Processes", frameon=True, fontsize=10)
plt.grid(axis='x', linestyle=":", alpha=0.6, linewidth=0.8)
plt.gca().spines["top"].set_visible(False)
plt.gca().spines["right"].set_visible(False)
plt.show()
```

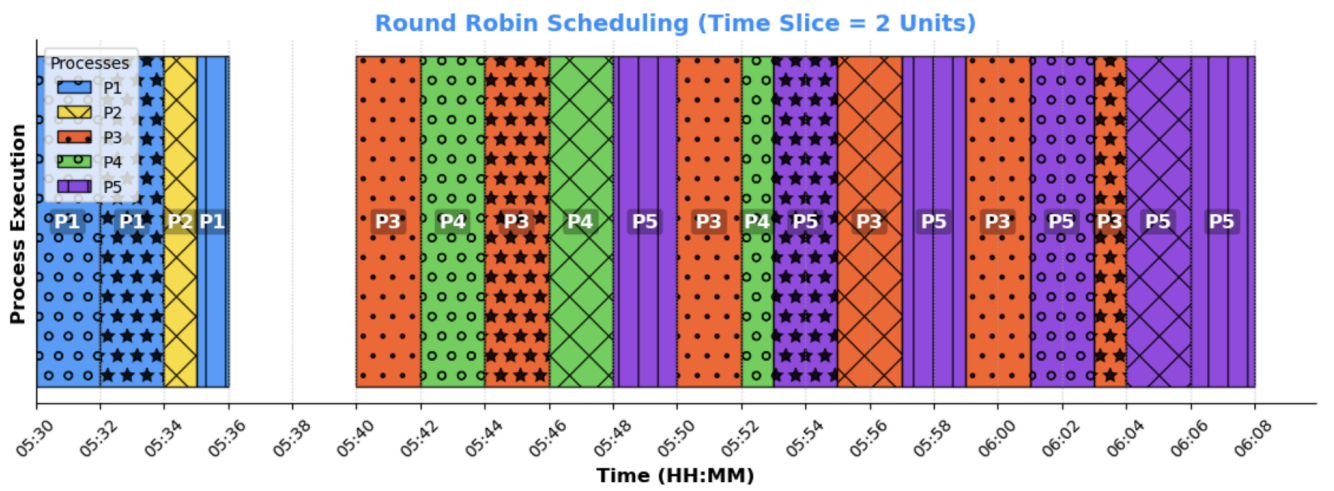


4. Time-Sharing System (Time Slice = 2 Units)

```
import matplotlib.pyplot as plt
from collections import deque

processes = [
    ("P1", 5 * 60 + 30, 5),
    ("P2", 5 * 60 + 33, 1),
    ("P3", 5 * 60 + 40, 11),
    ("P4", 5 * 60 + 42, 5),
    ("P5", 5 * 60 + 45, 12)
]

time_slice = 2
queue = deque()
```

```
remaining_time = {p[0]: p[2] for p in processes}
completed_processes = []
gantt_rr = []
current_time = min(p[1] for p in processes)
arrival_index = 0

colors = {
   "P1": "#1E90FF",
   "P2": "#FFD700",
   "P3": "#FF4500",
   "P4": "#32CD32",
   "P5": "#8A2BE2"
}

hatch_patterns = ['o', '*', 'x', '|', '.']

while len(completed_processes) < len(processes):
   for i in range(arrival_index, len(processes)):
      if processes[i][1] <= current_time:
         queue.append(processes[i][0])
         arrival_index += 1
      else:
         break

   if queue:
      process_id = queue.popleft()
      execution_time = min(time_slice, remaining_time[process_id])
      gantt_rr.append((process_id, current_time, current_time + execution_time))
      remaining_time[process_id] -= execution_time
      current_time += execution_time

      for i in range(arrival_index, len(processes)):
         if processes[i][1] <= current_time and processes[i][0] not in queue and processes[i][0]
not in completed_processes:
            queue.append(processes[i][0])
            arrival_index += 1

      if remaining_time[process_id] > 0:
         queue.append(process_id)
      else:
```

```
            completed_processes.append(process_id)
    else:
        current_time += 1

def minutes_to_time(minutes):
    hours = minutes // 60
    mins = minutes % 60
    return f"{hours:02d}:{mins:02d}"

time_range = range(gantt_rr[0][1], gantt_rr[-1][2] + 1, 1)
time_labels = [minutes_to_time(t) for t in time_range]

plt.figure(figsize=(14, 4))

for i, (process_id, start, end) in enumerate(gantt_rr):
    plt.barh(y=0, width=end-start, left=start, height=0.5,
             color=colors[process_id], edgecolor="black", alpha=0.85,
             hatch=hatch_patterns[i % len(hatch_patterns)], label=process_id if process_id not in
[p[0] for p in gantt_rr[:i]] else "")

    plt.text((start + end) / 2, 0, process_id, ha="center", va="center",
             fontsize=12, color="white", fontweight="bold",
             bbox=dict(facecolor='black', alpha=0.3, edgecolor='none', boxstyle="round,pad=0.2"))

plt.xticks(time_range[::2], time_labels[::2], rotation=45, fontsize=10)
plt.xlabel("Time (HH:MM)", fontsize=12, fontweight="bold")
plt.ylabel("Process Execution", fontsize=12, fontweight="bold")
plt.title("Round Robin Scheduling (Time Slice = 2 Units)", fontsize=14, fontweight="bold",
color="#1E90FF")

plt.yticks([])
plt.legend(loc="upper left", title="Processes", frameon=True, fontsize=10)
plt.grid(axis='x', linestyle=":", alpha=0.6, linewidth=0.8)
plt.gca().spines["top"].set_visible(False)
plt.gca().spines["right"].set_visible(False)
plt.show()
```

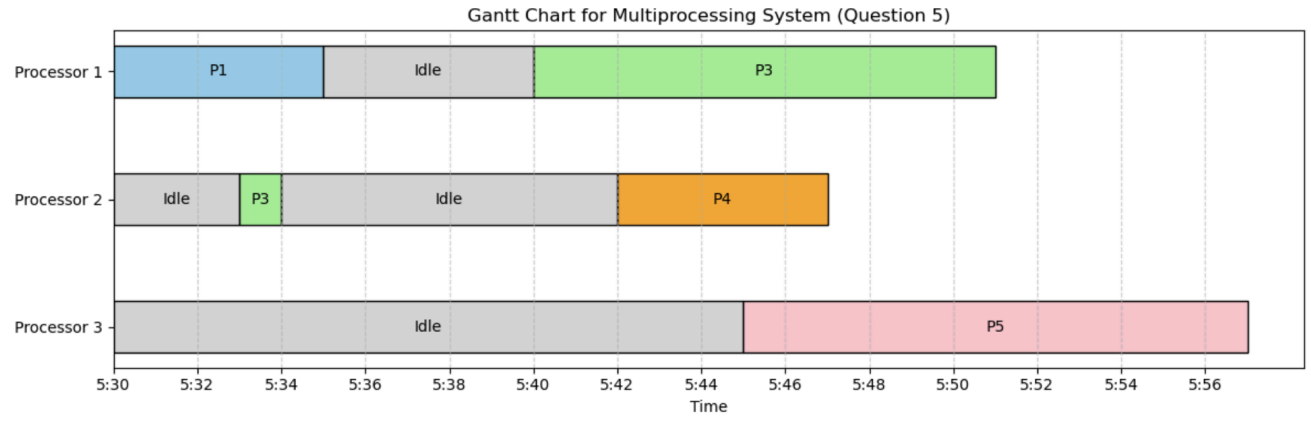


5. Multiprocessing System (3 Processors)

```
import matplotlib.pyplot as plt

def format_time(minutes):
    base_hour = 5
    base_min = 30
    total_min = base_min + minutes
    hour = base_hour + total_min // 60
    minute = total_min % 60
    return f"{hour}:{minute:02d}"

# Define segments for each processor (times in minutes relative to 5:30 PM)
segments = {
    "Processor 1": [
        (0, 5, "P1"),       # 5:30 - 5:35
        (5, 10, "Idle"),    # 5:35 - 5:40
        (10, 21, "P3")      # 5:40 - 5:51
    ],
    "Processor 2": [
        (0, 3, "Idle"),     # 5:30 - 5:33
        (3, 4, "P3"),       # 5:33 - 5:34
        (4, 12, "Idle"),    # 5:34 - 5:42
        (12, 17, "P4")      # 5:42 - 5:47
    ],
```

```
    "Processor 3": [
        (0, 15, "Idle"),     # 5:30 - 5:45
        (15, 27, "P5")       # 5:45 - 5:57
    ]
}

# Map processor names to y positions (Processor 1 at the top)
processor_names = list(segments.keys())
y_positions = list(reversed(range(len(processor_names))))
y_mapping = {proc: y for proc, y in zip(processor_names, y_positions)}

# Define colors for different labels
color_mapping = {
    "P1": "skyblue",
    "P3": "lightgreen",
    "P4": "orange",
    "P5": "pink",
    "Idle": "lightgray"
}

# Create the plot
fig, ax = plt.subplots(figsize=(12, 4))
bar_height = 0.4

# Plot each segment for every processor
for proc, seg_list in segments.items():
    y = y_mapping[proc]
    for start, end, label in seg_list:
        duration = end - start
        color = color_mapping.get(label, "gray")
        ax.barh(y, duration, left=start, height=bar_height, align='center',
                color=color, edgecolor='black')
        ax.text(start + duration / 2, y, label, ha='center', va='center',
                fontsize=10, color='black')

# Set y-axis labels with processor names
ax.set_yticks(list(y_mapping.values()))
ax.set_yticklabels(list(y_mapping.keys()))

# Set x-axis ticks every 2 minutes from 0 to 27 minutes
```

```
ticks = list(range(0, 28, 2))
ax.set_xticks(ticks)
tick_labels = [format_time(tick) for tick in ticks]
ax.set_xticklabels(tick_labels)

ax.set_xlabel("Time")
ax.set_title("Gantt Chart for Multiprocessing System (Question 5)")
ax.grid(True, axis='x', linestyle='--', alpha=0.6)

plt.tight_layout()
plt.show()
```

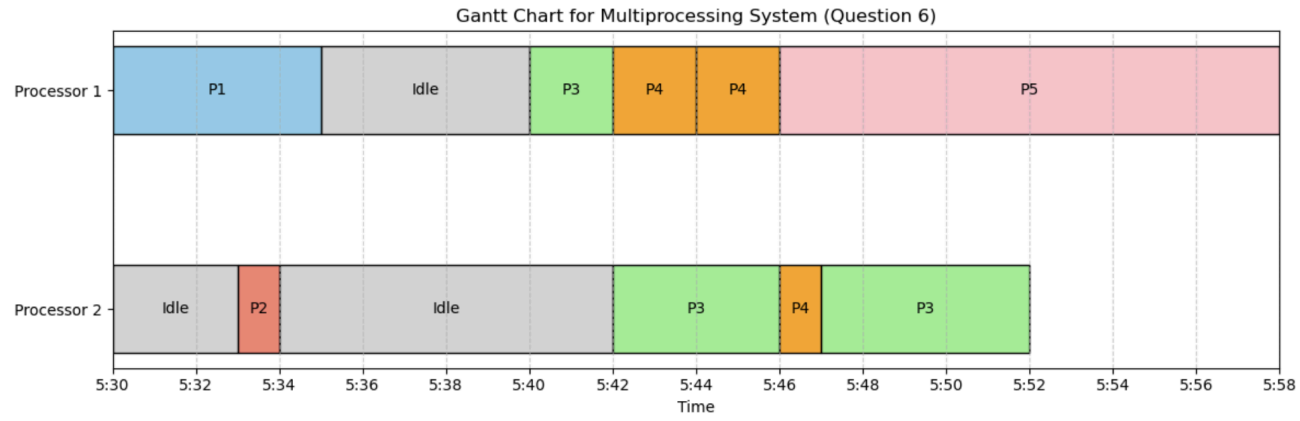


6. Multiprocessing Time-Sharing System (2 Processors, Time Slice = 2 units)

```
import matplotlib.pyplot as plt

def format_time(minutes):
    base_hour = 5
    base_min = 30
    total_min = base_min + minutes
    hour = base_hour + total_min // 60
    minute = total_min % 60
    return f"{hour}:{minute:02d}"

# Define segments for each processor in minutes relative to 5:30 PM
segments = {
    "Processor 1": [
        (0, 5, "P1"),       # 5:30 - 5:35
```

```
        (5, 10, "Idle"),     # 5:35 - 5:40
        (10, 12, "P3"),      # 5:40 - 5:42
        (12, 14, "P4"),      # 5:42 - 5:44
        (14, 16, "P4"),      # 5:44 - 5:46
        (16, 28, "P5")       # 5:46 - 5:58
    ],
    "Processor 2": [
        (0, 3, "Idle"),      # 5:30 - 5:33
        (3, 4, "P2"),        # 5:33 - 5:34
        (4, 12, "Idle"),     # 5:34 - 5:42
        (12, 16, "P3"),      # 5:42 - 5:46
        (16, 17, "P4"),      # 5:46 - 5:47
        (17, 22, "P3")       # 5:47 - 5:52
    ]
}

# Map processor names to y positions (Processor 1 on top)
processor_names = list(segments.keys())
y_positions = list(reversed(range(len(processor_names))))
y_mapping = {proc: y for proc, y in zip(processor_names, y_positions)}

# Define colors for each label
color_mapping = {
    "P1": "skyblue",
    "P2": "salmon",
    "P3": "lightgreen",
    "P4": "orange",
    "P5": "pink",
    "Idle": "lightgray"
}

# Create the plot
fig, ax = plt.subplots(figsize=(12, 4))
bar_height = 0.4

# Plot each segment for every processor
for proc, seg_list in segments.items():
    y = y_mapping[proc]
    for start, end, label in seg_list:
        duration = end - start
```

```
        color = color_mapping.get(label, "gray")
        ax.barh(y, duration, left=start, height=bar_height, align='center',
                color=color, edgecolor='black')
        ax.text(start + duration/2, y, label, ha='center', va='center',
                fontsize=10, color='black')

# Set y-axis labels with processor names
ax.set_yticks(list(y_mapping.values()))
ax.set_yticklabels(list(y_mapping.keys()))

# Set x-axis ticks every 2 minutes from 0 to 28 and label them with full times
ticks = list(range(0, 29, 2))
tick_labels = [format_time(tick) for tick in ticks]
ax.set_xticks(ticks)
ax.set_xticklabels(tick_labels)
ax.set_xlim(0, 28)
ax.set_xlabel("Time")
ax.set_title("Gantt Chart for Multiprocessing System (Question 6)")
ax.grid(True, axis='x', linestyle='--', alpha=0.6)

plt.tight_layout()
plt.show()
```



Assignment 4

NAVNEET YADAV
21105127 (ECE)

PID	Arrival time	Burst time
P1	5:30 PM	5 unit
P2	5:33 PM	1 unit
P3	5:40 PM	11 unit
P4	5:42 PM	5 unit
P5	5:45 PM	12 unit

Ans1. Multiprogramming system (FCFS)

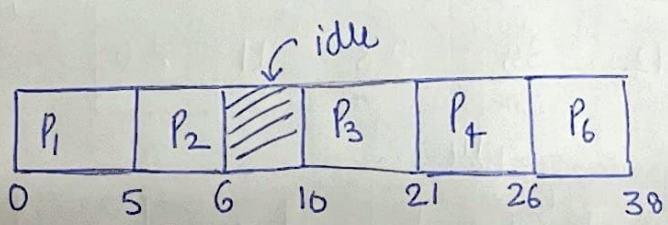
PID	AT	BT	CT	TAT	WT	RT
P_1	0	5	5	5	0	0
P_2	3	1	6	3	2	2
P_3	10	11	21	11	0	0
P_4	12	5	26	14	9	9
P_5	15	12	38	23	11	11

$\text{Avg TAT} = \frac{56}{5}$

11.2

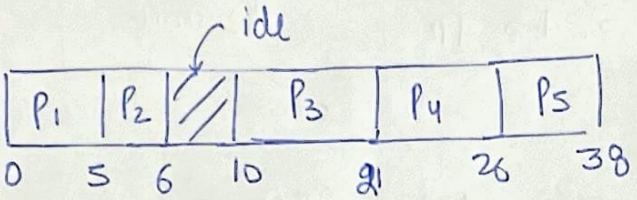
$\text{Avg WT} = \frac{22}{5}$

4.4



Ans2. SJF

PID	AT	BT	CT	TAT	WT	RT
P_1	0	5	5	5	0	0
P_2	3	1	6	3	2	2
P_3	10	11	21	11	0	0
P_4	12	5	26	14	9	9
P_5	15	12	38	23	11	11



Avg TAT = $\frac{56}{5}$ = 11.2

Avg WT = $\frac{22}{5}$ = 4.4

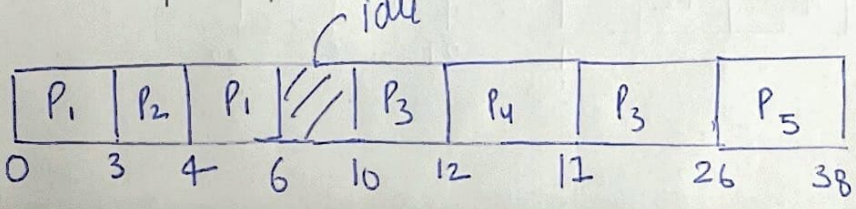
Ans3. SJF in preemptive mode

PID	AT	BT	CT	TAT	WT	RT
P_1	0	5	6	6	1	0
P_2	3	1	4	1	0	0
P_3	10	11	26	16	5	0
P_4	12	5	17	5	0	0
P_5	15	12	38	23	11	26

Avg TAT = $\frac{51}{5}$ = 10.2

Avg WT = $\frac{17}{5}$ = 3.4

Avg RT = 5.2



$P_1 \rightarrow 2$

$P_2 \rightarrow 1$

$P_3 \rightarrow 9$

$P_4 \rightarrow 8$

$P_5 \rightarrow 12$

Ans4.

Round Robin:
(Time sharing system) with 2 unit time

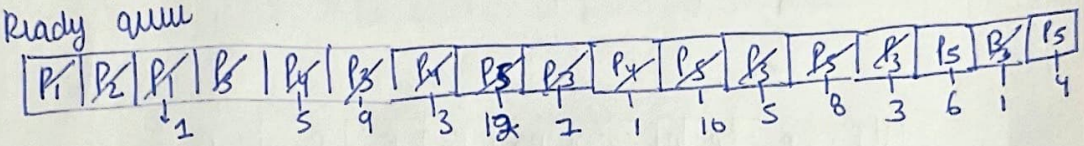
PID	AT	BT	CT	TAT	WT	RT
P_1	0	5	6	6	1	0
P_2	3	1	5	2	1	1
P_3	10	11	34	24	13	0
P_4	12	5	23	11	6	0
P_5	15	12	38	23	11	3

Avg TAT = $\frac{66}{5}$ = 13.2

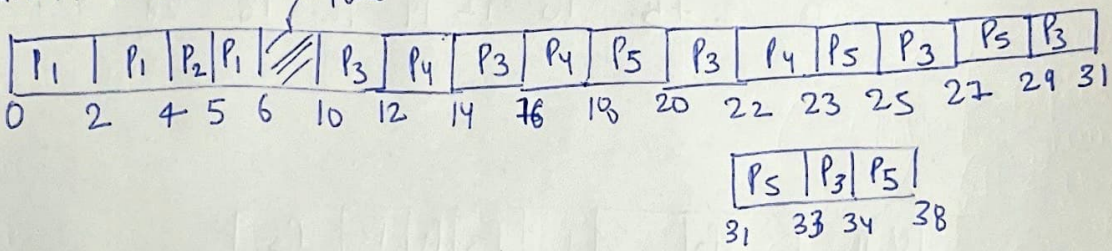
Avg WT = 32/5 = 6.4

Avg RT = 4/5 = 0.8

Ready queue



GANTT chart



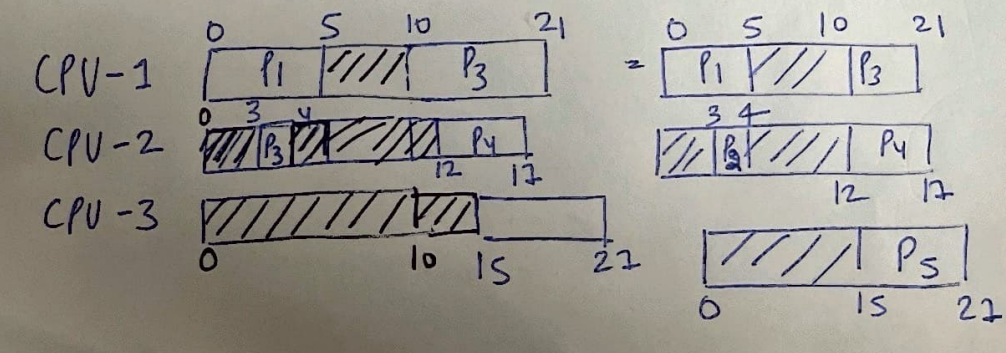
Ans5. Multiprocessing system with 3 processor

PID	AT	BT	CT	TAT	WT	RT
P_1	0	5	5	5	0	0
P_2	3	1	4	1	0	0
P_3	10	11	21	11	0	0
P_4	12	5	17	5	0	0
P_5	15	12	27	12	0	0

Avg WT = 0

Avg RT = 0

Avg TAT = $\frac{34}{5}$ = 6.8



Ans6. Multiprocessor : 2 processor

Time slice 2unit of Time

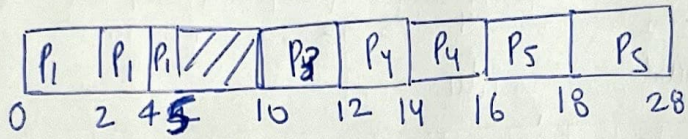
PID	AT	BT	CT	TAT	WT	RT
P_1	0	5	5	5	0	0
P_2	3	1	4	1	0	0
P_3	10	11	22	12	1	0
P_4	12	5	17	5	0	0
P_5	15	12	20	13	1	1

Avg TAT = $\frac{36}{5}$ = 7.2

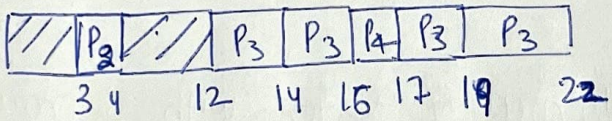
Avg WT = $\frac{2}{5}$ = 0.4

Avg RT = $\frac{1}{5}$ = 0.2

CPU I



CPU II



Ready queue

P5	P4	P5	P1	P3	P5	P4	P3	P5	P3
1	1	1	1	1	1	1	1	1	1
11	5	9	3	7	12	1	5	10	3