# Assignment 5
# 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

**Navneet Yadav**
**21105127**
**ECE**

# Assignment 5

**Problem:**

**Write an implementation of below CPU scheduling algorithms. Take user input for arrival time/ burst time / priority and produce completion time, waiting time, turn around time, average waiting time, average turnaround time, and**
**Gantt charts.**

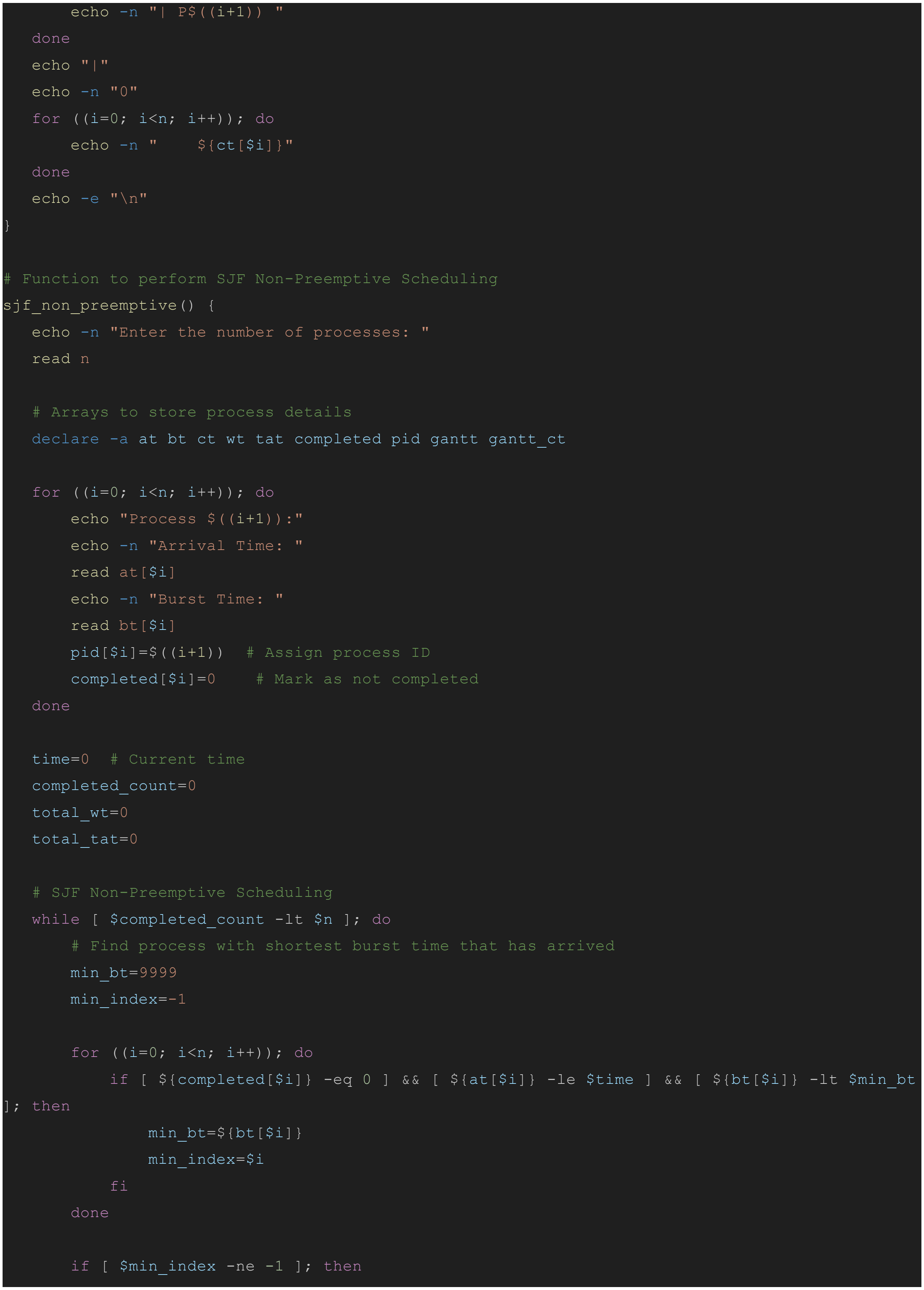
1. **FCFS Scheduling**
2. **SJF Scheduling (Non-Preemptive and Preemptive)**
3. **Non- Preemptive Priority Scheduling**
4. **Round Robin Scheduling**

**Make four different algorithms and finally combine all four in a single**

**Answer:**

```
#!/bin/bash

# Function to perform FCFS Scheduling
fcfs_scheduling() {
   echo -n "Enter the number of processes: "
   read n

   # Arrays to store process details
   declare -a at bt ct wt tat

   for ((i=0; i<n; i++)); do
       echo "Process $((i+1)):"
       echo -n "Arrival Time: "
       read at[$i]
       echo -n "Burst Time: "
       read bt[$i]
   done

   # Sort processes by arrival time
   for ((i=0; i<n-1; i++)); do
       for ((j=0; j<n-i-1; j++)); do
           if [ ${at[$j]} -gt ${at[$((j+1))]} ]; then
```

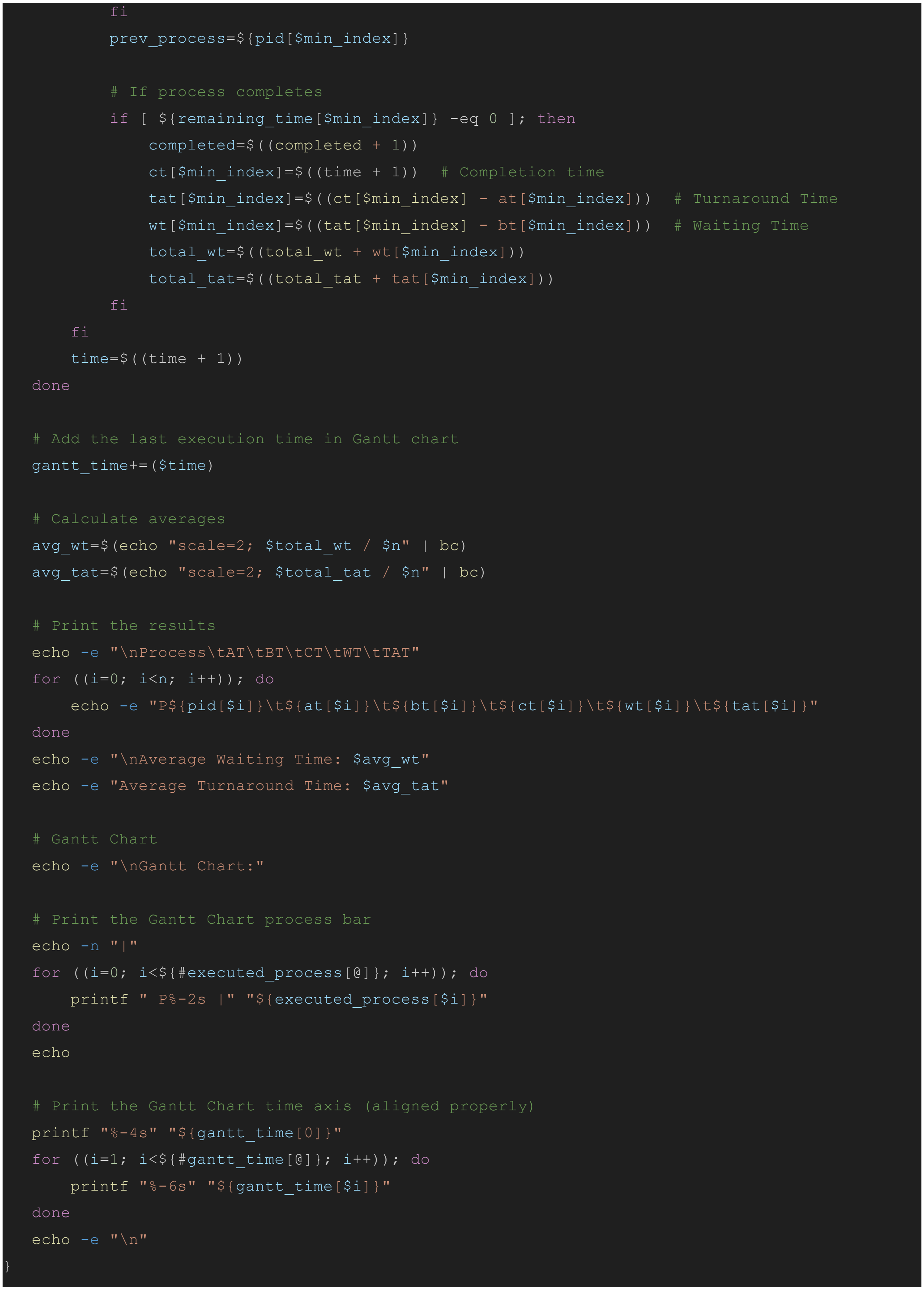
```
            # Swap arrival time
            temp=${at[$j]}
            at[$j]=${at[$((j+1))]}
            at[$((j+1))]=$temp

            # Swap burst time
            temp=${bt[$j]}
            bt[$j]=${bt[$((j+1))]}
            bt[$((j+1))]=$temp
        fi
    done
done

# Calculate Completion Time, Turnaround Time, and Waiting Time
ct[0]=$((at[0] + bt[0]))
tat[0]=$((ct[0] - at[0]))
wt[0]=$((tat[0] - bt[0]))

total_wt=${wt[0]}
total_tat=${tat[0]}

for ((i=1; i<n; i++)); do
    if [ ${ct[$((i-1))]} -lt ${at[$i]} ]; then
        ct[$i]=$((at[$i] + bt[$i]))
    else
        ct[$i]=$((ct[$((i-1))] + bt[$i]))
    fi
    tat[$i]=$((ct[$i] - at[$i]))
    wt[$i]=$((tat[$i] - bt[$i]))

    total_wt=$((total_wt + wt[$i]))
    total_tat=$((total_tat + tat[$i]))
done

avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

# Print the results
echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
for ((i=0; i<n; i++)); do
    echo -e "P$((i+1))\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
done
echo -e "\nAverage Waiting Time: $avg_wt"
echo -e "Average Turnaround Time: $avg_tat"

# Gantt Chart
echo -e "\nGantt Chart:"
for ((i=0; i<n; i++)); do
```



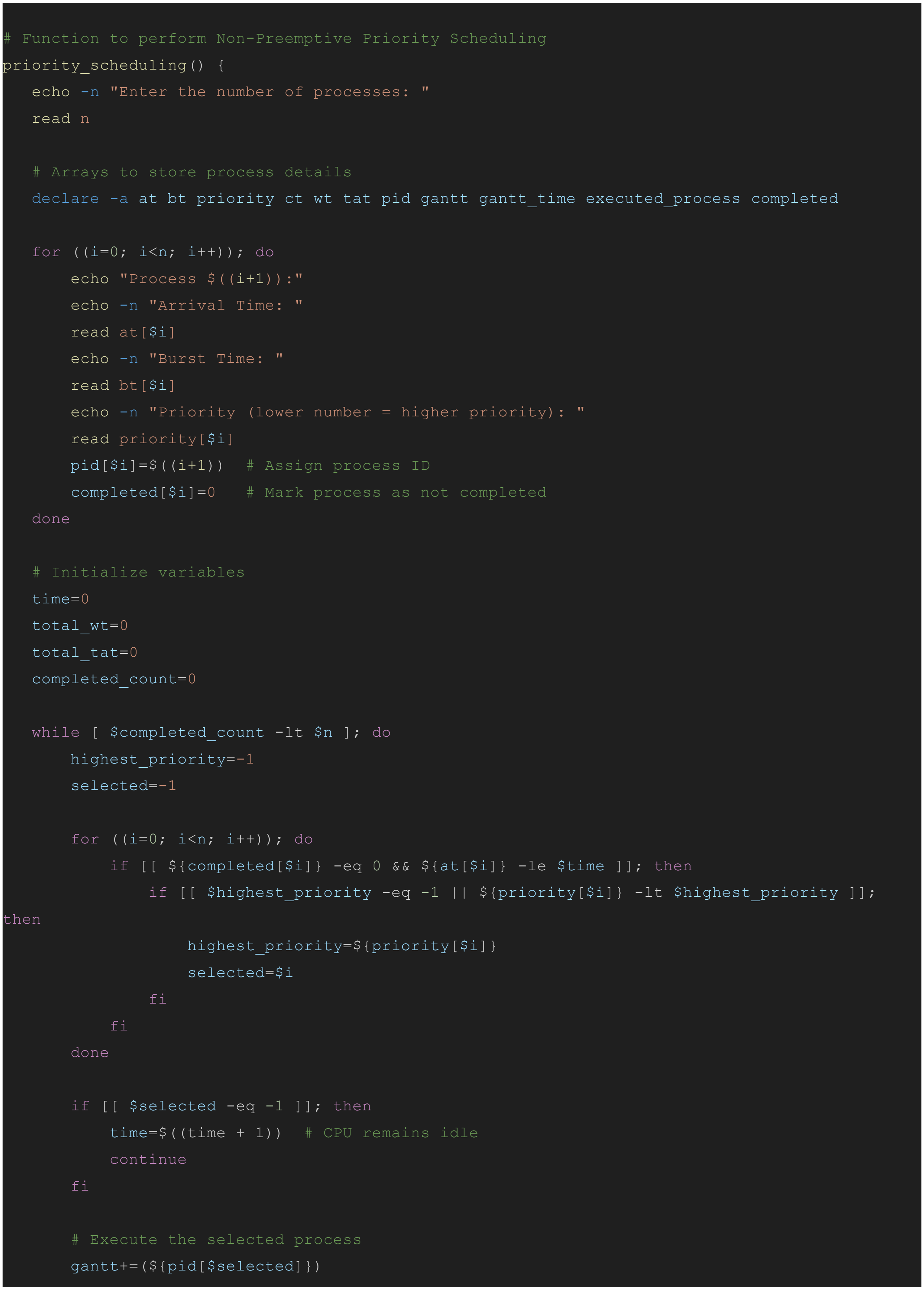
```
        echo -n "| P$((i+1)) "
    done
    echo "|"
    echo -n "0"
    for ((i=0; i<n; i++)); do
        echo -n "    ${ct[$i]}"
    done
    echo -e "\n"
}

# Function to perform SJF Non-Preemptive Scheduling
sjf_non_preemptive() {
    echo -n "Enter the number of processes: "
    read n

    # Arrays to store process details
    declare -a at bt ct wt tat completed pid gantt gantt_ct

    for ((i=0; i<n; i++)); do
        echo "Process $((i+1)):"
        echo -n "Arrival Time: "
        read at[$i]
        echo -n "Burst Time: "
        read bt[$i]
        pid[$i]=$((i+1))  # Assign process ID
        completed[$i]=0    # Mark as not completed
    done

    time=0  # Current time
    completed_count=0
    total_wt=0
    total_tat=0

    # SJF Non-Preemptive Scheduling
    while [ $completed_count -lt $n ]; do
        # Find process with shortest burst time that has arrived
        min_bt=9999
        min_index=-1

        for ((i=0; i<n; i++)); do
            if [ ${completed[$i]} -eq 0 ] && [ ${at[$i]} -le $time ] && [ ${bt[$i]} -lt $min_bt ]; then
                min_bt=${bt[$i]}
                min_index=$i
            fi
        done

        if [ $min_index -ne -1 ]; then
```

```
            # Calculate completion, turnaround, and waiting times
            time=$((time + bt[$min_index]))
            ct[$min_index]=$time
            tat[$min_index]=$((ct[$min_index] - at[$min_index]))
            wt[$min_index]=$((tat[$min_index] - bt[$min_index]))
            total_wt=$((total_wt + wt[$min_index]))
            total_tat=$((total_tat + tat[$min_index]))

            # Mark process as completed
            completed[$min_index]=1
            gantt[$completed_count]=${pid[$min_index]}  # Store process ID for Gantt chart
            gantt_ct[$completed_count]=${ct[$min_index]}  # Store completion time for Gantt
chart
            completed_count=$((completed_count + 1))
        else
            time=$((time + 1))  # Increment time if no process is ready
        fi
    done

    # Calculate averages
    avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
    avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

    # Print the results
    echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
    for ((i=0; i<n; i++)); do
        echo -e "P${pid[$i]}\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
    done
    echo -e "\nAverage Waiting Time: $avg_wt"
    echo -e "Average Turnaround Time: $avg_tat"

    # Gantt Chart
    echo -e "\nGantt Chart:"

    # Print the Gantt Chart process bar
    echo -n "|"
    for ((i=0; i<completed_count; i++)); do
        printf " P%-2s |" "${gantt[$i]}"
    done
    echo

    # Print the Gantt Chart time axis (aligned properly)
    printf "%-4s" "0"  # Start from time 0
    for ((i=0; i<completed_count; i++)); do
        printf "%-6s" "${gantt_ct[$i]}"
    done
    echo -e "\n"
}
```

```
# Function to perform SJF Preemptive (Shortest Remaining Time First - SRTF) Scheduling
sjf_preemptive() {
    echo -n "Enter the number of processes: "
    read n

    # Arrays to store process details
    declare -a at bt remaining_time ct wt tat pid gantt gantt_time executed_process

    for ((i=0; i<n; i++)); do
        echo "Process $((i+1)):"
        echo -n "Arrival Time: "
        read at[$i]
        echo -n "Burst Time: "
        read bt[$i]
        pid[$i]=$((i+1))  # Assign process ID
        remaining_time[$i]=${bt[$i]}  # Initially, remaining time = burst time
    done

    time=0  # Current time
    completed=0
    total_wt=0
    total_tat=0
    prev_process=-1  # Track previous process to record Gantt Chart changes

    # SJF Preemptive Scheduling (SRTF)
    while [ $completed -lt $n ]; do
        # Find the process with the shortest remaining time that has arrived
        min_rt=9999
        min_index=-1

        for ((i=0; i<n; i++)); do
            if [ ${remaining_time[$i]} -gt 0 ] && [ ${at[$i]} -le $time ] && [ ${remaining_time[$i]} -lt $min_rt ]; then
                min_rt=${remaining_time[$i]}
                min_index=$i
            fi
        done

        if [ $min_index -ne -1 ]; then
            # Execute the selected process for 1 time unit
            remaining_time[$min_index]=$((remaining_time[$min_index] - 1))
            gantt[$time]=${pid[$min_index]}  # Store process ID in Gantt chart

            # If a new process starts execution, mark the time
            if [ ${pid[$min_index]} -ne $prev_process ]; then
                gantt_time+=($time)
                executed_process+=(${pid[$min_index]})
```



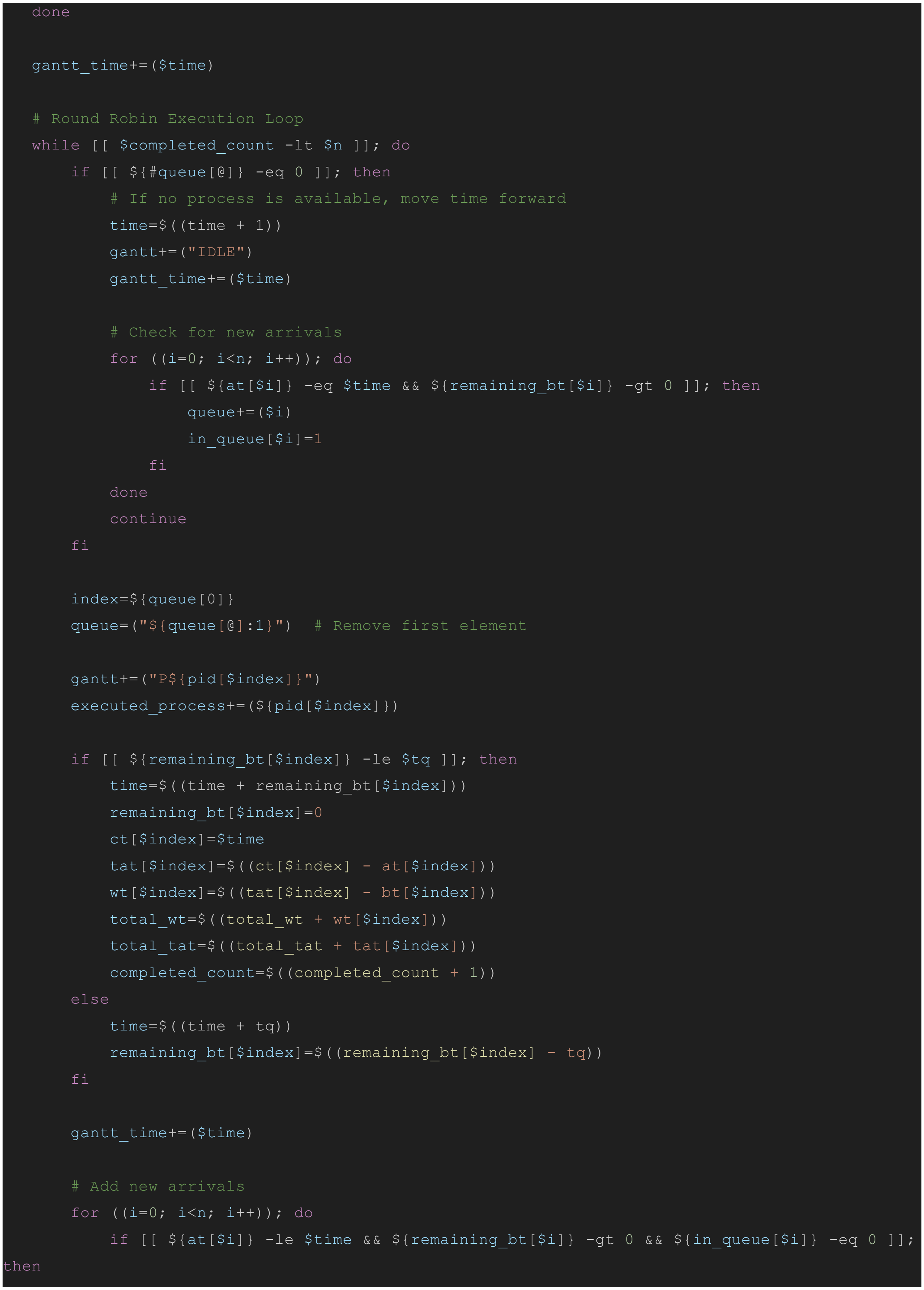
```
            fi
            prev_process=${pid[$min_index]}

            # If process completes
            if [ ${remaining_time[$min_index]} -eq 0 ]; then
                completed=$((completed + 1))
                ct[$min_index]=$((time + 1))  # Completion time
                tat[$min_index]=$((ct[$min_index] - at[$min_index]))  # Turnaround Time
                wt[$min_index]=$((tat[$min_index] - bt[$min_index]))  # Waiting Time
                total_wt=$((total_wt + wt[$min_index]))
                total_tat=$((total_tat + tat[$min_index]))
            fi
        fi
        time=$((time + 1))
    done

    # Add the last execution time in Gantt chart
    gantt_time+=($time)

    # Calculate averages
    avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
    avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

    # Print the results
    echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
    for ((i=0; i<n; i++)); do
        echo -e "P${pid[$i]}\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
    done
    echo -e "\nAverage Waiting Time: $avg_wt"
    echo -e "Average Turnaround Time: $avg_tat"

    # Gantt Chart
    echo -e "\nGantt Chart:"

    # Print the Gantt Chart process bar
    echo -n "|"
    for ((i=0; i<${#executed_process[@]}; i++)); do
        printf " P%-2s |" "${executed_process[$i]}"
    done
    echo

    # Print the Gantt Chart time axis (aligned properly)
    printf "%-4s" "${gantt_time[0]}"
    for ((i=1; i<${#gantt_time[@]}; i++)); do
        printf "%-6s" "${gantt_time[$i]}"
    done
    echo -e "\n"
}
```



```
# Function to perform Non-Preemptive Priority Scheduling
priority_scheduling() {
   echo -n "Enter the number of processes: "
   read n

   # Arrays to store process details
   declare -a at bt priority ct wt tat pid gantt gantt_time executed_process completed

   for ((i=0; i<n; i++)); do
       echo "Process $((i+1)):"
       echo -n "Arrival Time: "
       read at[$i]
       echo -n "Burst Time: "
       read bt[$i]
       echo -n "Priority (lower number = higher priority): "
       read priority[$i]
       pid[$i]=$((i+1))  # Assign process ID
       completed[$i]=0   # Mark process as not completed
   done

   # Initialize variables
   time=0
   total_wt=0
   total_tat=0
   completed_count=0

   while [ $completed_count -lt $n ]; do
       highest_priority=-1
       selected=-1

       for ((i=0; i<n; i++)); do
           if [[ ${completed[$i]} -eq 0 && ${at[$i]} -le $time ]]; then
               if [[ $highest_priority -eq -1 || ${priority[$i]} -lt $highest_priority ]];
then
                   highest_priority=${priority[$i]}
                   selected=$i
               fi
           fi
       done

       if [[ $selected -eq -1 ]]; then
           time=$((time + 1))  # CPU remains idle
           continue
       fi

       # Execute the selected process
       gantt+=(${pid[$selected]})
```

```
        gantt_time+=($time)
        executed_process+=(${pid[$selected]})

        ct[$selected]=$((time + bt[$selected]))  # Completion time
        tat[$selected]=$((ct[$selected] - at[$selected]))  # Turnaround Time
        wt[$selected]=$((tat[$selected] - bt[$selected]))  # Waiting Time
        total_wt=$((total_wt + wt[$selected]))
        total_tat=$((total_tat + tat[$selected]))

        time=${ct[$selected]}  # Move time forward
        completed[$selected]=1  # Mark as completed
        completed_count=$((completed_count + 1))
    done

    gantt_time+=($time)  # Add last execution time in Gantt chart

    # Calculate averages
    avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
    avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

    # Print the results
    echo -e "\nProcess\tAT\tBT\tPr\tCT\tWT\tTAT"
    for ((i=0; i<n; i++)); do
        echo -e
"P${pid[$i]}\t${at[$i]}\t${bt[$i]}\t${priority[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
    done
    echo -e "\nAverage Waiting Time: $avg_wt"
    echo -e "Average Turnaround Time: $avg_tat"

    # Gantt Chart
    echo -e "\nGantt Chart:"

    # Print the Gantt Chart process bar
    echo -n "|"
    for ((i=0; i<${#executed_process[@]}; i++)); do
        printf " P%-2s |" "${executed_process[$i]}"
    done
    echo

    # Print the Gantt Chart time axis (aligned properly)
    printf "%-4s" "${gantt_time[0]}"
    for ((i=1; i<${#gantt_time[@]}; i++)); do
        printf "%-6s" "${gantt_time[$i]}"
    done
    echo -e "\n"
}

# Function for Round Robin Scheduling
```

```
round_robin() {
    echo -n "Enter the number of processes: "
    read n

    # Arrays to store process data
    declare -a at bt ct wt tat remaining_bt pid gantt gantt_time executed_process

    for ((i=0; i<n; i++)); do
        echo "Process $((i+1)):"
        echo -n "Arrival Time: "
        read at[$i]
        echo -n "Burst Time: "
        read bt[$i]
        pid[$i]=$((i+1))  # Assign process ID
        remaining_bt[$i]=${bt[$i]}  # Initialize remaining burst time
    done

    echo -n "Enter Time Quantum: "
    read tq

    # Initialize variables
    time=0
    total_wt=0
    total_tat=0
    queue=()
    completed_count=0
    in_queue=()

    # Function to check if process is in queue
    is_in_queue() {
        local search=$1
        for p in "${queue[@]}"; do
            if [[ $p -eq $search ]]; then
                return 0  # Found
            fi
        done
        return 1  # Not found
    }

    # Add processes that arrive at time = 0
    for ((i=0; i<n; i++)); do
        in_queue[$i]=0  # Initialize queue tracking
    done
    for ((i=0; i<n; i++)); do
        if [[ ${at[$i]} -eq 0 ]]; then
            queue+=($i)
            in_queue[$i]=1
        fi
```

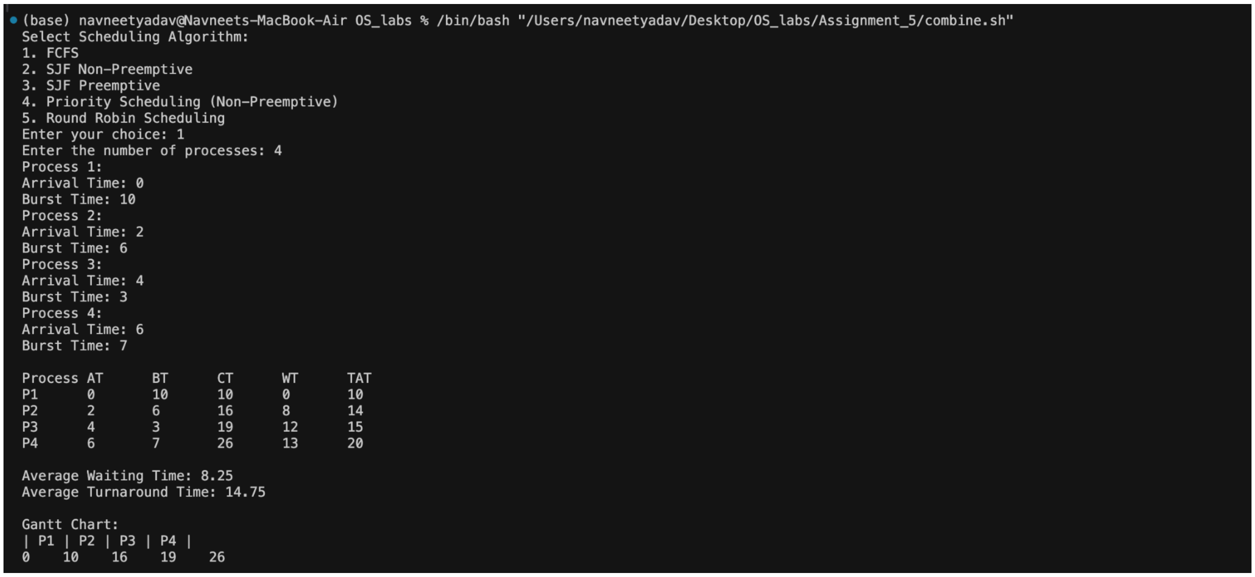


```
    done

    gantt_time+=($time)

    # Round Robin Execution Loop
    while [[ $completed_count -lt $n ]]; do
        if [[ ${#queue[@]} -eq 0 ]]; then
            # If no process is available, move time forward
            time=$((time + 1))
            gantt+=("IDLE")
            gantt_time+=($time)

            # Check for new arrivals
            for ((i=0; i<n; i++)); do
                if [[ ${at[$i]} -eq $time && ${remaining_bt[$i]} -gt 0 ]]; then
                    queue+=($i)
                    in_queue[$i]=1
                fi
            done
            continue
        fi

        index=${queue[0]}
        queue=("${queue[@]:1}")  # Remove first element

        gantt+=("P${pid[$index]}")
        executed_process+=(${pid[$index]})

        if [[ ${remaining_bt[$index]} -le $tq ]]; then
            time=$((time + remaining_bt[$index]))
            remaining_bt[$index]=0
            ct[$index]=$time
            tat[$index]=$((ct[$index] - at[$index]))
            wt[$index]=$((tat[$index] - bt[$index]))
            total_wt=$((total_wt + wt[$index]))
            total_tat=$((total_tat + tat[$index]))
            completed_count=$((completed_count + 1))
        else
            time=$((time + tq))
            remaining_bt[$index]=$((remaining_bt[$index] - tq))
        fi

        gantt_time+=($time)

        # Add new arrivals
        for ((i=0; i<n; i++)); do
            if [[ ${at[$i]} -le $time && ${remaining_bt[$i]} -gt 0 && ${in_queue[$i]} -eq 0 ]];
then
```

```
                queue+=($i)
                in_queue[$i]=1
            fi
        done

        # Reinsert process at the end if it still needs execution
        if [[ ${remaining_bt[$index]} -gt 0 ]]; then
            queue+=($index)
        fi
    done

    # Calculate averages
    avg_wt=$(echo "scale=2; $total_wt / $n" | bc)
    avg_tat=$(echo "scale=2; $total_tat / $n" | bc)

    # Print Process Table
    echo -e "\nProcess\tAT\tBT\tCT\tWT\tTAT"
    for ((i=0; i<n; i++)); do
        echo -e "P${pid[$i]}\t${at[$i]}\t${bt[$i]}\t${ct[$i]}\t${wt[$i]}\t${tat[$i]}"
    done
    echo -e "\nAverage Waiting Time: $avg_wt"
    echo -e "Average Turnaround Time: $avg_tat"

    # Print Gantt Chart
    echo -e "\nGantt Chart:"
    echo -n "|"
    for ((i=0; i<${#gantt[@]}; i++)); do
        printf " %-5s |" "${gantt[$i]}"
    done
    echo

    # Print the Gantt Chart time axis
    printf "%-5s" "${gantt_time[0]}"
    for ((i=1; i<${#gantt_time[@]}; i++)); do
        printf "%-7s" "${gantt_time[$i]}"
    done
    echo -e "\n"
}



# Ask user which algorithm to use
echo "Select Scheduling Algorithm:"
echo "1. FCFS"
echo "2. SJF Non-Preemptive"
echo "3. SJF Preemptive"
echo "4. Priority Scheduling (Non-Preemptive)"
echo "5. Round Robin Scheduling "
echo -n "Enter your choice: "
read choice

case $choice in
    1) fcfs_scheduling ;;
    2) sjf_non_preemptive ;;
    3) sjf_preemptive ;;
    4) priority_scheduling;;
    5) round_robin;;
    *) echo "Invalid choice!" ;;
esac
```

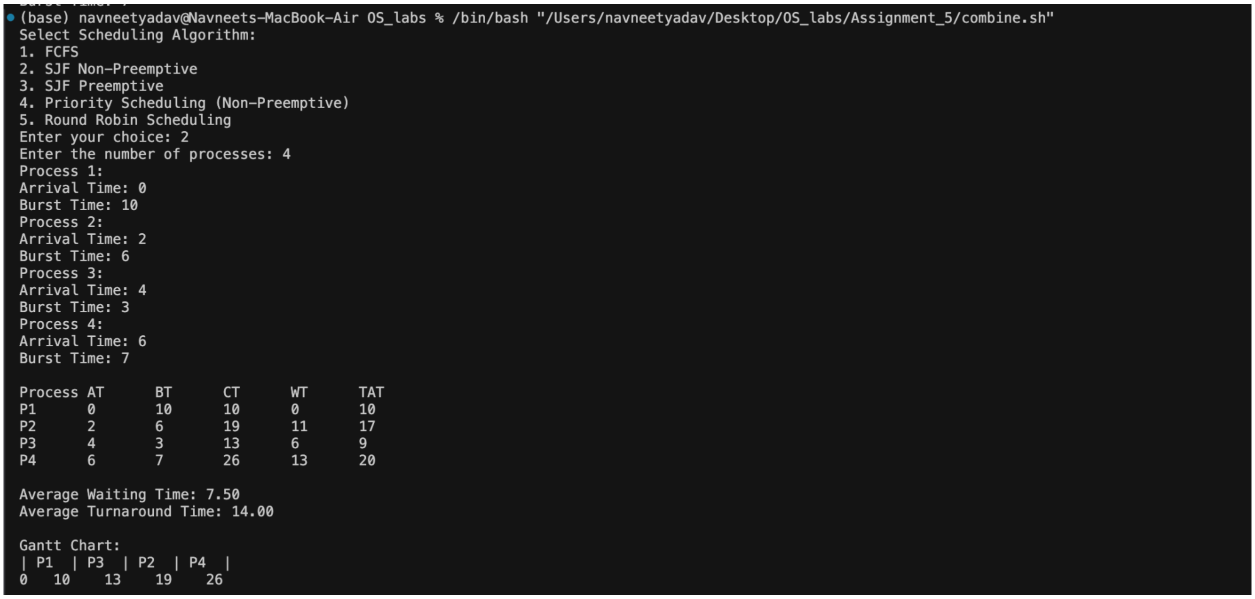
## Output:

1. **FCFS Scheduling**



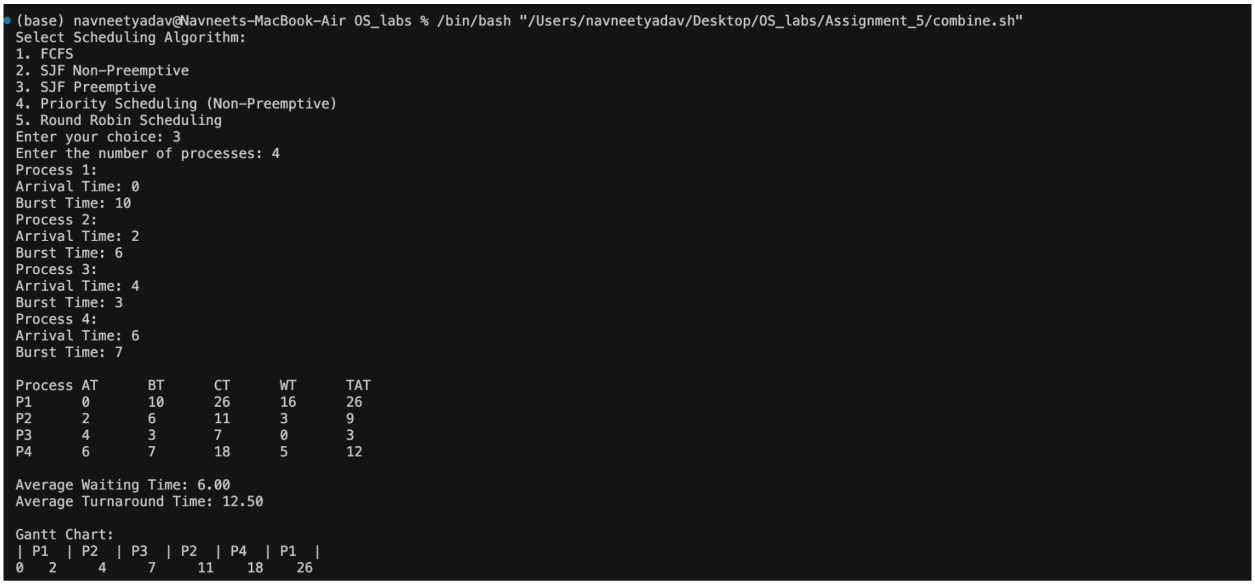
```
(base) navneetyadav@Navneets-MacBook-Air OS_labs % /bin/bash "/Users/navneetyadav/Desktop/OS_labs/Assignment_5/combine.sh"
Select Scheduling Algorithm:
1. FCFS
2. SJF Non-Preemptive
3. SJF Preemptive
4. Priority Scheduling (Non-Preemptive)
5. Round Robin Scheduling
Enter your choice: 1
Enter the number of processes: 4
Process 1:
Arrival Time: 0
Burst Time: 10
Process 2:
Arrival Time: 2
Burst Time: 6
Process 3:
Arrival Time: 4
Burst Time: 3
Process 4:
Arrival Time: 6
Burst Time: 7

Process AT      BT      CT      WT      TAT
P1      0       10      10      0       10
P2      2       6       16      8       14
P3      4       3       19      12      15
P4      6       7       26      13      20

Average Waiting Time: 8.25
Average Turnaround Time: 14.75

Gantt Chart:
| P1 | P2 | P3 | P4 |
0    10    16    19    26
```

## 2. SJF Scheduling (Non-Preemptive)



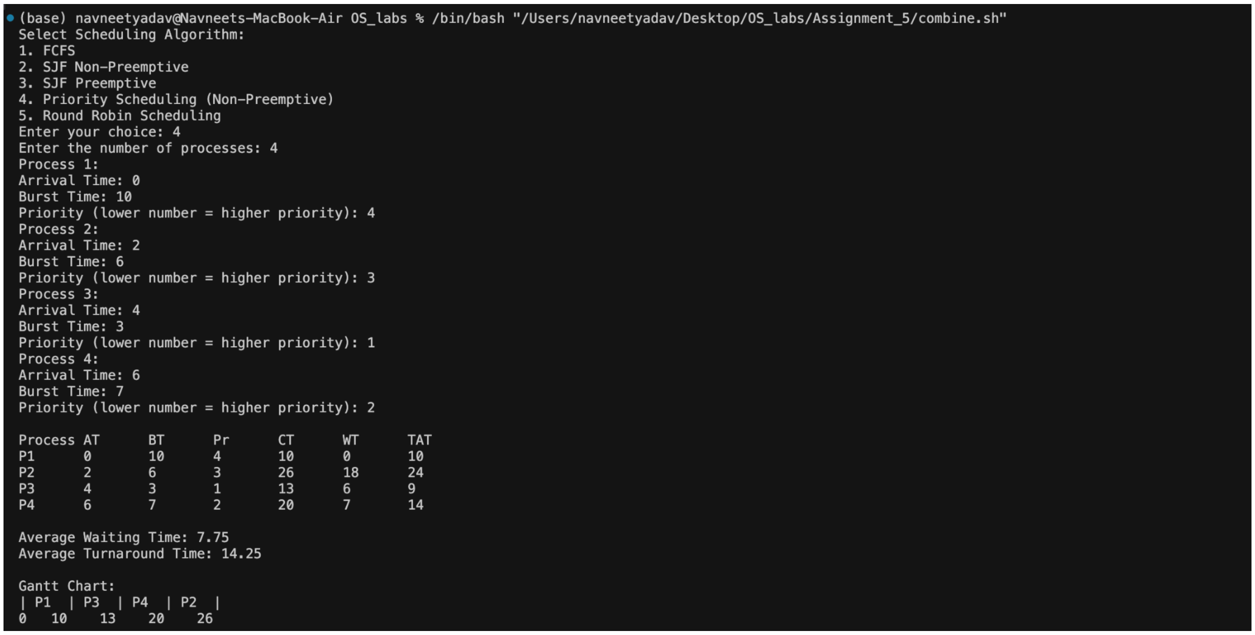
```
(base) navneetyadav@Navneets-MacBook-Air OS_labs % /bin/bash "/Users/navneetyadav/Desktop/OS_labs/Assignment_5/combine.sh"
Select Scheduling Algorithm:
1. FCFS
2. SJF Non-Preemptive
3. SJF Preemptive
4. Priority Scheduling (Non-Preemptive)
5. Round Robin Scheduling
Enter your choice: 2
Enter the number of processes: 4
Process 1:
Arrival Time: 0
Burst Time: 10
Process 2:
Arrival Time: 2
Burst Time: 6
Process 3:
Arrival Time: 4
Burst Time: 3
Process 4:
Arrival Time: 6
Burst Time: 7

Process AT      BT      CT      WT      TAT
P1      0       10      10      0       10
P2      2       6       19      11      17
P3      4       3       13      6       9
P4      6       7       26      13      20

Average Waiting Time: 7.50
Average Turnaround Time: 14.00

Gantt Chart:
| P1  | P3  | P2  | P4  |
0   10    13    19    26
```

## 3. SJF Scheduling (Preemptive)



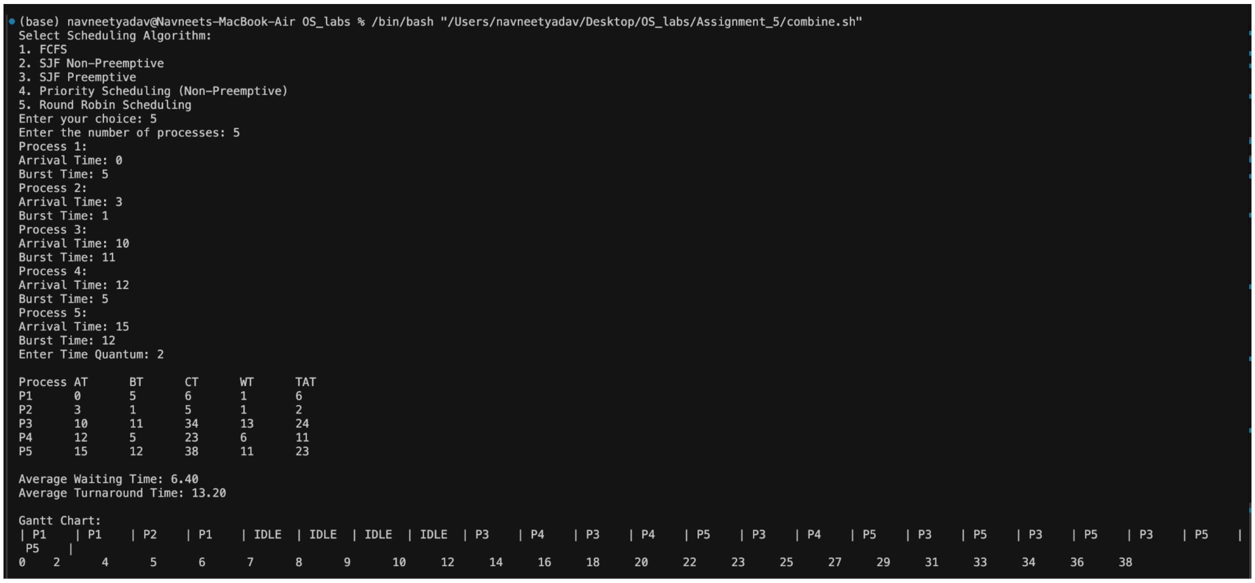
```
(base) navneetyadav@Navneets-MacBook-Air OS_labs % /bin/bash "/Users/navneetyadav/Desktop/OS_labs/Assignment_5/combine.sh"
Select Scheduling Algorithm:
1. FCFS
2. SJF Non-Preemptive
3. SJF Preemptive
4. Priority Scheduling (Non-Preemptive)
5. Round Robin Scheduling
Enter your choice: 3
Enter the number of processes: 4
Process 1:
Arrival Time: 0
Burst Time: 10
Process 2:
Arrival Time: 2
Burst Time: 6
Process 3:
Arrival Time: 4
Burst Time: 3
Process 4:
Arrival Time: 6
Burst Time: 7

Process AT      BT      CT      WT      TAT
P1      0       10      26      16      26
P2      2       6       11      3       9
P3      4       3       7       0       3
P4      6       7       18      5       12

Average Waiting Time: 6.00
Average Turnaround Time: 12.50

Gantt Chart:
| P1  | P2  | P3  | P2  | P4  | P1  |
0   2     4     7     11    18    26
```

## 4. Non- Preemptive Priority Scheduling



```
(base) navneetyadav@Navneets-MacBook-Air OS_labs % /bin/bash "/Users/navneetyadav/Desktop/OS_labs/Assignment_5/combine.sh"
Select Scheduling Algorithm:
1. FCFS
2. SJF Non-Preemptive
3. SJF Preemptive
4. Priority Scheduling (Non-Preemptive)
5. Round Robin Scheduling
Enter your choice: 4
Enter the number of processes: 4
Process 1:
Arrival Time: 0
Burst Time: 10
Priority (lower number = higher priority): 4
Process 2:
Arrival Time: 2
Burst Time: 6
Priority (lower number = higher priority): 3
Process 3:
Arrival Time: 4
Burst Time: 3
Priority (lower number = higher priority): 1
Process 4:
Arrival Time: 6
Burst Time: 7
Priority (lower number = higher priority): 2

Process AT      BT      Pr      CT      WT      TAT
P1      0       10      4       10      0       10
P2      2       6       3       26      18      24
P3      4       3       1       13      6       9
P4      6       7       2       20      7       14

Average Waiting Time: 7.75
Average Turnaround Time: 14.25

Gantt Chart:
| P1  | P3  | P4  | P2  |
0   10    13    20    26
```

## 5. Round Robin Scheduling



```
(base) navneetyadav@Navneets-MacBook-Air OS_labs % /bin/bash "/Users/navneetyadav/Desktop/OS_labs/Assignment_5/combine.sh"
Select Scheduling Algorithm:
1. FCFS
2. SJF Non-Preemptive
3. SJF Preemptive
4. Priority Scheduling (Non-Preemptive)
5. Round Robin Scheduling
Enter your choice: 5
Enter the number of processes: 5
Process 1:
Arrival Time: 0
Burst Time: 5
Process 2:
Arrival Time: 3
Burst Time: 1
Process 3:
Arrival Time: 10
Burst Time: 11
Process 4:
Arrival Time: 12
Burst Time: 5
Process 5:
Arrival Time: 15
Burst Time: 12
Enter Time Quantum: 2

Process AT      BT      CT      WT      TAT
P1      0       5       6       1       6
P2      3       1       5       1       2
P3      10      11      34      13      24
P4      12      5       23      6       11
P5      15      12      38      11      23

Average Waiting Time: 6.40
Average Turnaround Time: 13.20

Gantt Chart:
| P1    | P1    | P2    | P1    | IDLE  | IDLE  | IDLE  | IDLE  | P3    | P4    | P3    | P4    | P5    | P3    | P4    | P5    | P3    | P5    | P3    | P5    | P3    | P5    |
 P5    |
0    2      4      5      6      7      8      9      10     12     14     16     18     20     22     23     25     27     29     31     33     34     36     38
```