Name-Abhi Mehta
Reg No.-221080001

Experiment-03

Aim: The implementation of below CPU scheduling algorithms in Shell scripting. 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 scripts and finally combine all four executable scripts in a single script by taking case conditions to execute each.

Theory:

1] FCFS Scheduling
Mode of execution- Default Non-Preemptive
Criteria for selection- Arrival Time

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ cat Scheduling1.sh
#!/bin/bash

echo "Enter the number of processes:"
read n

declare -a arrival
declare -a burst

declare -a completion
declare -a waiting
declare -a turnaround

declare -A process_numbers

for ((i = 1; i <= n; i++)); do
  echo "Enter arrival time for Process $i:"
  read arrival[$i]
  echo "Enter burst time for Process $i:"
  read burst[$i]
  process_numbers[$i]=$i
done

# Sort processes based on arrival time
for ((i = 1; i <= n; i++)); do
  for ((j = i + 1; j <= n; j++)); do
    if [ ${arrival[$i]} -gt ${arrival[$j]} ]; then
      temp=${arrival[$i]}
      arrival[$i]=${arrival[$j]}
      arrival[$j]=$temp

      temp=${burst[$i]}
      burst[$i]=${burst[$j]}
      burst[$j]=$temp

      temp=${process_numbers[$i]}
      process_numbers[$i]=${process_numbers[$j]}
      process_numbers[$j]=$temp
    fi
  done
done

current_time=${arrival[1]}
remaining_time=( "${burst[@]}" )
processed=( false false false false )

echo -e "\nSJF Preemptive Scheduling:"
echo -e "Time\tProcess"

gantt_chart=""

while true; do
  min_burst_time=99999
  selected_process=-1
```

```
  for ((i = 1; i <= n; i++)); do
    if [[ ${arrival[$i]} -le $current_time && ${remaining_time[$i]} -lt $min_burst_time && ${processed[$i]} == false ]]; then
      min_burst_time=${remaining_time[$i]}
      selected_process=$i
    fi
  done

  if [ $selected_process -eq -1 ]; then
    done=true
    for ((i = 1; i <= n; i++)); do
      if [[ ${processed[$i]} == false ]]; then
        done=false
        break
      fi
    done
    if $done; then
      break
    fi
    gantt_chart+="IDLE $((${arrival[$i]} - current_time)) |"
    current_time=${arrival[$i]}
  else
    gantt_chart+="P${process_numbers[$selected_process]} |"
    remaining_time[$selected_process]=$((remaining_time[$selected_process] - 1))
    current_time=$((current_time + 1))
    if [ ${remaining_time[$selected_process]} -eq 0 ]; then
      processed[$selected_process]=true
    fi
  fi
done

echo "$gantt_chart"

total_waiting=0
total_turnaround=0

for ((i = 1; i <= n; i++)); do
  turnaround_time=$((current_time - arrival[$i]))
  waiting_time=$((turnaround_time - burst[$i]))
  total_waiting=$((total_waiting + waiting_time))
  total_turnaround=$((total_turnaround + turnaround_time))
done

avg_waiting_time=$(echo "scale=2; $total_waiting / $n" | bc)
avg_turnaround_time=$(echo "scale=2; $total_turnaround / $n" | bc)

echo -e "\nAverage Waiting Time: $avg_waiting_time"
echo "Average Turnaround Time: $avg_turnaround_time"
```

Output-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ chmod +x Scheduling1.sh
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ ./Scheduling1.sh
Enter the number of processes:
4
Enter arrival time for Process 1:
0
Enter burst time for Process 1:
2
Enter arrival time for Process 2:
1
Enter burst time for Process 2:
2
Enter arrival time for Process 3:
5
Enter burst time for Process 3:
3
Enter arrival time for Process 4:
6
Enter burst time for Process 4:
4

SJF Preemptive Scheduling:
Time    Process
P1 |P1 |P2 |P2 |P2 |P3 |P3 |P3 |P3 |

Average Waiting Time: 3.25
Average Turnaround Time: 6.00
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$
```

2] SJF SchedulingMode of execution- Non-Preemptive / Preemptive Criteria for selection- Burst Time Script (non-preemptive) –

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ cat scheduling2.sh
#!/bin/bash

echo "Enter the number of processes:"
read n

declare -a arrival
declare -a burst

declare -a completion
declare -a waiting
declare -a turnaround

declare -A process_numbers

for ((i = 1; i <= n; i++)); do
  echo "Enter arrival time for Process $i:"
  read arrival[$i]
  echo "Enter burst time for Process $i:"
  read burst[$i]
  process_numbers[$i]=$i
done

# Sort processes based on burst time
for ((i = 1; i <= n; i++)); do
  for ((j = i + 1; j <= n; j++)); do
    if [ ${burst[$i]} -gt ${burst[$j]} ]; then
      temp=${arrival[$i]}
      arrival[$i]=${arrival[$j]}
      arrival[$j]=$temp

      temp=${burst[$i]}
      burst[$i]=${burst[$j]}
      burst[$j]=$temp

      temp=${process_numbers[$i]}
      process_numbers[$i]=${process_numbers[$j]}
      process_numbers[$j]=$temp
    fi
  done
done

# Calculate completion, waiting, and turnaround times
completion[0]=0
waiting[0]=0
turnaround[0]=0

for ((i = 1; i <= n; i++)); do
  completion[$i]=$((completion[$i - 1] + burst[$i]))
  waiting[$i]=$((completion[$i - 1]))
  turnaround[$i]=$((completion[$i] - arrival[$i]))
done

echo -e "\nSJF Non-Preemptive Scheduling:"
```

```
echo -e "Process\tArrival Time\tBurst Time\tCompletion Time\tWaiting Time\tTurnaround Time"

for ((i = 1; i <= n; i++)); do
  original_process_number=${process_numbers[$i]}
  echo -e "$original_process_number\t${arrival[$i]}\t\t${burst[$i]}\t\t${completion[$i]}\t\t${waiting[$i]}\t\t${turnaround[$i]}"
done

total_waiting=0
total_turnaround=0

for ((i = 1; i <= n; i++)); do
  total_waiting=$((total_waiting + waiting[$i]))
  total_turnaround=$((total_turnaround + turnaround[$i]))
done

avg_waiting_time=$(echo "scale=2; $total_waiting / $n" | bc)
avg_turnaround_time=$(echo "scale=2; $total_turnaround / $n" | bc)

echo -e "\nAverage Waiting Time: $avg_waiting_time"
echo "Average Turnaround Time: $avg_turnaround_time"

echo -e "\nGenerating Gantt Chart..."
echo -n "0 "
for ((i = 1; i <= n; i++)); do
  original_process_number=${process_numbers[$i]}
  echo -n "| P$original_process_number $(printf "%${burst[$i]}s" | tr ' ' '-') T${completion[$i]} "
done
echo "|"
```

Output-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ chmod +x scheduling2.sh
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ ./scheduling2.sh
Enter the number of processes:
4
Enter arrival time for Process 1:
1
Enter burst time for Process 1:
3
Enter arrival time for Process 2:
2
Enter burst time for Process 2:
4
Enter arrival time for Process 3:
1
Enter burst time for Process 3:
2
Enter arrival time for Process 4:
4
Enter burst time for Process 4:
4

SJF Non-Preemptive Scheduling:
Process Arrival Time    Burst Time      Completion Time Waiting Time    Turnaround
3       1               2               2               0               1
1       1               3               5               2               4
2       2               4               9               5               7
4       4               4               13              9               9

Average Waiting Time: 4.00
Average Turnaround Time: 5.25

Generating Gantt Chart...
0 | P3 -- T2 | P1 --- T5 | P2 ---- T9 | P4 ---- T13 |
```

Script (preemptive) –

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ cat scheduling3.sh
#!/bin/bash

echo "Enter the number of processes:"
read n

declare -a arrival
declare -a burst
declare -a remaining_burst
declare -a completion
declare -a waiting
declare -a turnaround
declare -a processed
declare -A process_numbers

# Function to find the process with the shortest remaining burst time
findShortestJob() {
    local min_burst=99999
    local shortest_job=-1

    for ((i = 1; i <= n; i++)); do
        if [[ ${arrival[$i]} -le $current_time && ${remaining_burst[$i]} -lt $min_burst && ${processed[$i]} == false ]]; then
            min_burst=${remaining_burst[$i]}
            shortest_job=$i
        fi
    done

    echo $shortest_job
}

# Function to calculate average waiting and turnaround times
calculateAverages() {
    local total_waiting=0
    local total_turnaround=0

    for ((i = 1; i <= n; i++)); do
        local turnaround_time=$((completion[$i] - arrival[$i]))
        local waiting_time=$((turnaround_time - burst[$i]))
        total_waiting=$((total_waiting + waiting_time))
        total_turnaround=$((total_turnaround + turnaround_time))
    done

    local avg_waiting_time=$(echo "scale=2; $total_waiting / $n" | bc)
    local avg_turnaround_time=$(echo "scale=2; $total_turnaround / $n" | bc)

    echo "$avg_waiting_time"
    echo "$avg_turnaround_time"
}

# Input arrival and burst times for processes
for ((i = 1; i <= n; i++)); do
    echo "Enter arrival time for Process $i:"
    read arrival[$i]
```

```
    echo "Enter burst time for Process $i:"
    read burst[$i]
    remaining_burst[$i]=${burst[$i]}
    process_numbers[$i]=$i
    processed[$i]=false
done

# Sort processes based on arrival time
for ((i = 1; i <= n; i++)); do
    for ((j = i + 1; j <= n; j++)); do
        if [ ${arrival[$i]} -gt ${arrival[$j]} ]; then
            temp=${arrival[$i]}
            arrival[$i]=${arrival[$j]}
            arrival[$j]=$temp

            temp=${burst[$i]}
            burst[$i]=${burst[$j]}
            burst[$j]=$temp

            temp=${remaining_burst[$i]}
            remaining_burst[$i]=${remaining_burst[$j]}
            remaining_burst[$j]=$temp

            temp=${process_numbers[$i]}
            process_numbers[$i]=${process_numbers[$j]}
            process_numbers[$j]=$temp
        fi
    done
done

current_time=${arrival[1]}
gantt_chart=""

echo -e "\nSJF Preemptive Scheduling:"
echo -e "Time\tProcess"

# Main loop for scheduling
while true; do
    shortest_job=$(findShortestJob)

    if [ $shortest_job -eq -1 ]; then
        break
    fi

    gantt_chart+="P${process_numbers[$shortest_job]} |"
    current_time=$((current_time + 1))
    remaining_burst[$shortest_job]=$((remaining_burst[$shortest_job] - 1))

    if [ ${remaining_burst[$shortest_job]} -eq 0 ]; then
        processed[$shortest_job]=true
        completion[$shortest_job]=$current_time
    fi
done
```

```
echo "$gantt_chart"

avg_waiting_time=$(calculateAverages)
echo -e "\nAverage Waiting Time and average turnaround time: $avg_waiting_time"
```

Output-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ ./scheduling3.sh
Enter the number of processes:
4
Enter arrival time for Process 1:
0
Enter burst time for Process 1:
5
Enter arrival time for Process 2:
1
Enter burst time for Process 2:
3
Enter arrival time for Process 3:
2
Enter burst time for Process 3:
4
Enter arrival time for Process 4:
4
Enter burst time for Process 4:
1

SJF Preemptive Scheduling:
Time    Process
P1 |P2 |P2 |P2 |P4 |P1 |P1 |P1 |P1 |P3 |P3 |P3 |P3 |

Average Waiting Time and average turnaround time: 2.75
6.00
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$
```

3] Non-Preemptive Priority SchedulingMode of execution- Non-Preemptive Criteria for selection- Arrival Time and Priority Script-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ cat scheduling4.sh
#!/bin/bash

echo "Enter the number of processes:"
read n

declare -a arrival
declare -a burst
declare -a priority
declare -a completion
declare -a waiting
declare -a turnaround
declare -a processed
declare -a gantt

# Function to find the process with the highest priority
findHighestPriority() {
    local highest_priority=99999
    local highest_priority_process=-1

    for ((i = 0; i < n; i++)); do
        if [[ ${arrival[$i]} -le $current_time && ${processed[$i]} == false && ${priority[$i]} -lt $highest_priority ]]; then
            highest_priority=${priority[$i]}
            highest_priority_process=$i
        fi
    done

    echo $highest_priority_process
}

# Function to calculate completion time, waiting time, and turnaround time
calculateTimes() {
    local current_time=0
    local index=0

    while true; do
        all_processes_done=1
        highest_priority_process=$(findHighestPriority)
        if [[ $highest_priority_process -eq -1 ]]; then
            break
        fi
        gantt[$index]=$((highest_priority_process + 1))  # Adjust index to start from 1
        index=$((index + 1))
        current_time=$((current_time + burst[$highest_priority_process]))
        completion[$highest_priority_process]=$current_time
        waiting[$highest_priority_process]=$((current_time - arrival[$highest_priority_process] - burst[$highest_priority_process]))
        turnaround[$highest_priority_process]=$((current_time - arrival[$highest_priority_process]))
        processed[$highest_priority_process]=true
    done
}
```

```
# Function to calculate average waiting time and average turnaround time
calculateAverages() {
    local total_waiting=0
    local total_turnaround=0

    for ((i = 0; i < n; i++)); do
        total_waiting=$((total_waiting + waiting[$i]))
        total_turnaround=$((total_turnaround + turnaround[$i]))
    done

    local avg_waiting=$(echo "scale=2; $total_waiting / $n" | bc)
    local avg_turnaround=$(echo "scale=2; $total_turnaround / $n" | bc)

    echo "$avg_waiting"
    echo "$avg_turnaround"
}

# Input arrival time, burst time, and priority for each process
for ((i = 0; i < n; i++)); do
    echo "Enter arrival time for Process $((i + 1)):"
    read arrival[$i]
    echo "Enter burst time for Process $((i + 1)):"
    read burst[$i]
    echo "Enter priority for Process $((i + 1)):"
    read priority[$i]
    processed[$i]=false
done

calculateTimes

echo -e "\nNon-Preemptive Priority Scheduling:"
echo -e "Process\tArrival Time\tBurst Time\tPriority\tCompletion Time\tWaiting Time\tTurnaround Time"

for ((i = 0; i < n; i++)); do
    echo -e "$((i + 1))\t${arrival[$i]}\t\t${burst[$i]}\t\t${priority[$i]}\t\t${completion[$i]}\t\t${waiting[$i]}\t\t${turnaround[$i]}"
done

# Display Gantt chart
echo -e "\nGantt Chart:"
echo -n "0 "
for ((i = 0; i < ${#gantt[@]}; i++)); do
    echo -n "| P${gantt[$i]} "
done
echo -e "|"

# Display time below Gantt chart
echo -n "0 "
for ((i = 0; i < ${#gantt[@]}; i++)); do
    echo -n "  $((${completion[${gantt[$i]} - 1]}))  "
done
echo ""
```

```
# Display time below Gantt chart
echo -n "0 "
for ((i = 0; i < ${#gantt[@]}; i++)); do
    echo -n "  $((${completion[${gantt[$i]} - 1]}))  "
done
echo ""

avg=$(calculateAverages)
echo -e "\nAverage Waiting Time and Average Turnaround Time: $avg"
```

Output-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ chmod +x scheduling4.sh
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ ./scheduling4.sh
Enter the number of processes:
4
Enter arrival time for Process 1:
0
Enter burst time for Process 1:
5
Enter priority for Process 1:
4
Enter arrival time for Process 2:
1
Enter burst time for Process 2:
4
Enter priority for Process 2:
3
Enter arrival time for Process 3:
2
Enter burst time for Process 3:
2
Enter priority for Process 3:
2
Enter arrival time for Process 4:
4
Enter burst time for Process 4:
1
Enter priority for Process 4:
1

Non-Preemptive Priority Scheduling:
Process Arrival Time    Burst Time      Priority        Completion Time Waiting Time    Turnaround Time
1       0               5               4               5               0               5
2       1               4               3               12              7               11
3       2               2               2               8               4               6
4       4               1               1               6               1               2

Gantt Chart:
0 | P1 | P4 | P3 | P2 |
0   5    6    8    12

Average Waiting Time and Average Turnaround Time: 3.00
6.00
```

4] Round Robin SchedulingMode of execution- Preemptive Criteria for selection- Arrival Time and Time Quantum=2

Script-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ cat scheduling5.sh
#!/bin/bash

# Function to calculate average waiting and turnaround times
calculateAverages() {
    local total_waiting=0
    local total_turnaround=0

    for ((i = 1; i <= n; i++)); do
        local turnaround_time=$((completion[$i] - arrival[$i]))
        local waiting_time=$((turnaround_time - burst[$i]))
        total_waiting=$((total_waiting + waiting_time))
        total_turnaround=$((total_turnaround + turnaround_time))
    done

    local avg_waiting_time=$(echo "scale=2; $total_waiting / $n" | bc)
    local avg_turnaround_time=$(echo "scale=2; $total_turnaround / $n" | bc)

    echo "$avg_waiting_time"
    echo "$avg_turnaround_time"
}

# Input number of processes
echo "Enter the number of processes:"
read n

# Declare arrays
declare -a arrival
declare -a burst
declare -a remaining_burst
declare -a completion
declare -a waiting
declare -a turnaround
declare -a processed
declare -A process_numbers

# Input arrival and burst times for processes
for ((i = 1; i <= n; i++)); do
    echo "Enter arrival time for Process $i:"
    read arrival[$i]
    echo "Enter burst time for Process $i:"
    read burst[$i]
    remaining_burst[$i]=${burst[$i]}
    process_numbers[$i]=$i
    processed[$i]=false
done

# Input time quantum
echo "Enter the time quantum:"
read time_quantum

current_time=0
```

```
gantt_chart=""
echo -e "\nRound Robin Scheduling (Time Quantum: $time_quantum):"
echo -e "Time\tProcess\tBT\tCT\tTAT\tWT"

# Main loop for scheduling
while true; do
    all_processes_done=true
    for ((i = 1; i <= n; i++)); do
        if [[ ${remaining_burst[$i]} -gt 0 ]]; then
            all_processes_done=false
            if [[ ${remaining_burst[$i]} -gt $time_quantum ]]; then
                gantt_chart+="P${process_numbers[$i]} |"
                current_time=$((current_time + time_quantum))
                remaining_burst[$i]=$((remaining_burst[$i] - time_quantum))
            else
                gantt_chart+="P${process_numbers[$i]} |"
                current_time=$((current_time + remaining_burst[$i]))
                completion[$i]=$current_time
                remaining_burst[$i]=0
                processed[$i]=true
                printf "%d\t%d\t%d\t%d\t%d\t%d\n" "$current_time" "${process_numbers[$i]}" "${burst[$i]}" "${completion[$i]}" "$((completion[$i] - arrival[$i]))" "$((completion[$i] - arrival[$i
] - burst[$i]))"
            fi
        fi
    done
    if $all_processes_done; then
        break
    fi
done

echo "$gantt_chart"

# Calculate and display average waiting and turnaround times
avg_times=$(calculateAverages)
echo -e "\nAverage Waiting Time and Average Turnaround Time: $avg_times"
```

Output-

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ chmod +x scheduling5.sh
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ ./scheduling5.sh
Enter the number of processes:
4
Enter arrival time for Process 1:
0
Enter burst time for Process 1:
5
Enter arrival time for Process 2:
1
Enter burst time for Process 2:
4
Enter arrival time for Process 3:
2
Enter burst time for Process 3:
2
Enter arrival time for Process 4:
4
Enter burst time for Process 4:
1
Enter the time quantum:
2

Round Robin Scheduling (Time Quantum: 2):
Time    Process BT      CT      TAT     WT
6       3       2       6       4       2
7       4       1       7       3       2
11      2       4       11      10      6
12      1       5       12      12      7
P1 |P2 |P3 |P4 |P1 |P2 |P1 |

Average Waiting Time and Average Turnaround Time: 4.25
7.25
```

Combined-

```
#!/bin/bash

# Function for First-Come, First-Served (FCFS) Scheduling
fcfs_scheduling() {
    echo "Enter the number of processes:"
    read n

    # Arrays to store process information
    declare -a arrival
    declare -a burst
    declare -a completion
    declare -a waiting
    declare -a turnaround

    # Function to calculate completion time, waiting time, and turn around time
    calculate_times() {
        local i
        completion[0]=${burst[0]}
        waiting[0]=0
        turnaround[0]=${completion[0]}
        for ((i=1; i<$n; i++))
        do
            completion[$i]=$(( ${completion[$((i-1))]} + ${burst[$i]} ))
            waiting[$i]=$(( ${completion[$((i-1))]} - ${arrival[$i]} ))
            turnaround[$i]=$(( ${waiting[$i]} + ${burst[$i]} ))
        done
    }

    # Function to calculate average waiting time and average turnaround time
    calculate_averages() {
        local i total_waiting=0 total_turnaround=0
        for ((i=0; i<$n; i++))
        do
```

```
        turnaround[0]=${completion[0]}
        for ((i=1; i<$n; i++))
        do
            completion[$i]=$(( ${completion[$((i-1))]} + ${burst[$i]} ))
            waiting[$i]=$(( ${completion[$((i-1))]} - ${arrival[$i]} ))
            turnaround[$i]=$(( ${waiting[$i]} + ${burst[$i]} ))
        done
    }

    # Function to calculate average waiting time and average turnaround time
    calculate_averages() {
        local i total_waiting=0 total_turnaround=0
        for ((i=0; i<$n; i++))
        do
            total_waiting=$(( $total_waiting + ${waiting[$i]} ))
            total_turnaround=$(( $total_turnaround + ${turnaround[$i]} ))
        done
        avg_waiting=$(echo "scale=2; $total_waiting / $n" | bc)
        avg_turnaround=$(echo "scale=2; $total_turnaround / $n" | bc)
    }

    # Function to print Gantt chart
    print_gantt_chart() {
        local i j
        printf "\nGantt Chart:\n"
        printf "|"
        for ((i=0; i<$n; i++))
        do
            printf " P%d |" "$((i+1))"
        done
        printf "\n"
        printf "0"
        for ((i=0; i<$n; i++))
        do
```

```
            printf " P%d |" "$((i+1))"
        done
        printf "\n"
        printf "0"
        for ((i=0; i<$n; i++))
        do
            printf "     %d" "${completion[$i]}"
        done
        printf "\n"
    }

    # Main script

    # Input arrival time and burst time for each process
    for ((i=0; i<$n; i++))
    do
        echo "Enter arrival time for process $((i+1)):"
        read arrival[$i]
        echo "Enter burst time for process $((i+1)):"
        read burst[$i]
    done

    calculate_times
    calculate_averages

    echo "Process    Arrival Time    Burst Time    Completion Time    Waiting Time    Turnaround Time"
    for ((i=0; i<$n; i++))
    do
        echo "  $((i+1))             ${arrival[$i]}                ${burst[$i]}                ${completion[$i]}
            ${waiting[$i]}                ${turnaround[$i]}"
    done

    echo "Average Waiting Time: $avg_waiting"
    echo "Average Turnaround Time: $avg_turnaround"
```

```
    echo "Average Waiting Time: $avg_waiting"
    echo "Average Turnaround Time: $avg_turnaround"

    print_gantt_chart
}

# Function for Shortest Job First (SJF) Non-Preemptive Scheduling
sjf_non_preemptive() {
    echo "Enter the number of processes:"
    read n

    # Arrays to store process information
    declare -a arrival
    declare -a burst
    declare -a completion
    declare -a waiting
    declare -a turnaround
    declare -a processed

    # Function to calculate completion time, waiting time, and turn around time
    calculate_times() {
        local i j min_index min_burst
        for ((i=0; i<$n; i++))
        do
            min_index=-1
            min_burst=999999
            for ((j=0; j<$n; j++))
            do
                if [[ ${arrival[$j]} -le $current_time && ${burst[$j]} -lt $min_burst && ${processed[$j]} -eq 0 ]]; the
n
                    min_burst=${burst[$j]}
                    min_index=$j
                fi
            done
```

```
            ((i--))
        else
            completion[$min_index]=$(( $current_time + ${burst[$min_index]} ))
            waiting[$min_index]=$(( $current_time - ${arrival[$min_index]} ))
            turnaround[$min_index]=$(( ${waiting[$min_index]} + ${burst[$min_index]} ))
            current_time=${completion[$min_index]}
            processed[$min_index]=1
        fi
    done
}

# Function to calculate average waiting time and average turnaround time
calculate_averages() {
    local i total_waiting=0 total_turnaround=0
    for ((i=0; i<$n; i++))
    do
        total_waiting=$(( $total_waiting + ${waiting[$i]} ))
        total_turnaround=$(( $total_turnaround + ${turnaround[$i]} ))
    done
    avg_waiting=$(echo "scale=2; $total_waiting / $n" | bc)
    avg_turnaround=$(echo "scale=2; $total_turnaround / $n" | bc)
}

# Function to print Gantt chart
print_gantt_chart() {
    local i j
    printf "\nGantt Chart:\n"
    printf "|"
    for ((i=0; i<$n; i++))
    do
        if [[ ${processed[$i]} -eq 1 ]]; then
            printf " P%d |" "$((i+1))"
        fi
    done
```

```
    do
        if [[ ${processed[$i]} -eq 1 ]]; then
            printf "     %d" "${completion[$i]}"
        fi
    done
    printf "\n"
}

# Main script

# Input arrival time and burst time for each process
for ((i=0; i<$n; i++))
do
    echo "Enter arrival time for process $((i+1)):"
    read arrival[$i]
    echo "Enter burst time for process $((i+1)):"
    read burst[$i]
    processed[$i]=0
done

current_time=0
calculate_times
calculate_averages

echo "Process    Arrival Time    Burst Time    Completion Time    Waiting Time    Turnaround Time"
for ((i=0; i<$n; i++))
do
    echo "  $((i+1))             ${arrival[$i]}                ${burst[$i]}                ${completion[$i]}
        ${waiting[$i]}                ${turnaround[$i]}"
done

echo "Average Waiting Time: $avg_waiting"
echo "Average Turnaround Time: $avg_turnaround"
```

```
    # Input arrival time and burst time for each process
    for ((i=0; i<$n; i++))
    do
        echo "Enter arrival time for process $((i+1)):"
        read arrival[$i]
        echo "Enter burst time for process $((i+1)):"
        read burst[$i]
        remaining[$i]=${burst[$i]}
    done

    calculate_times
    calculate_averages

    echo "Process    Arrival Time    Burst Time    Completion Time    Waiting Time    Turnaround Time"
    for ((i=0; i<$n; i++))
    do
        echo "  $((i+1))             ${arrival[$i]}                ${burst[$i]}              ${completion[$i]}
          ${waiting[$i]}                ${turnaround[$i]}"
    done

    echo "Average Waiting Time: $avg_waiting"
    echo "Average Turnaround Time: $avg_turnaround"

    print_gantt_chart
}

# Main script

# Display menu
echo "Select the scheduling algorithm:"
echo "1. FCFS Scheduling"
echo "2. SJF Non-Preemptive Scheduling"
echo "3. SJF Preemptive Scheduling"
```

```
    print_gantt_chart
}

# Main script

# Display menu
echo "Select the scheduling algorithm:"
echo "1. FCFS Scheduling"
echo "2. SJF Non-Preemptive Scheduling"
echo "3. SJF Preemptive Scheduling"
echo "4. Priority Scheduling"
echo "5. Round Robin Scheduling"
echo "Enter your choice:"
read choice

# Perform action based on choice
case $choice in
    1) fcfs_scheduling ;;
    2) sjf_non_preemptive ;;
    3) sjf_preemptive ;;
    4) priority_scheduling ;;
    5) round_robin_scheduling ;;
    *) echo "Invalid choice";;
esac
```

```
abhi@abhi-VivoBook-ASUSLaptop-M3400QA-M3400QA:~$ ./combined.sh

Menu:
1. FCFS Scheduling
2. SJF Preemptive Scheduling
3. SJF Non-Preemptive Scheduling
4. Non-Preemptive Priority Scheduling
5. Round Robin Scheduling
6. Exit
Enter your choice:
```

Conclusion-To conclude we studied the CPU scheduling algorithms in this experiment which help to utilize CPU time properly.The choice of scheduling algorithm depends on system requirements, workload characteristics, and performance objectives. Each algorithm has its strengths and weaknesses, and selecting the most appropriate one requires considering these factors to achieve optimal system performance.