Assignment 5

Create and execute C programs for following CPU Scheduling Algorithms

1. First Come First Serve (FCFS)

```
#include <stdio.h>
void fcfs() {
   printf("\n--- First Come First Serve (FCFS) Scheduling ---\n");
   int n, i, j;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   int process[n], burst_time[n], waiting_time[n], turnaround_time[n];
   float avg_waiting_time = 0, avg_turnaround_time = 0;
   printf("\nEnter Burst Time for each process:\n");
   for(i = 0; i < n; i++) {
      process[i] = i + 1;
      printf("P%d: ", i + 1);
      scanf("%d", &burst_time[i]);
   }
   // Calculate waiting time
   waiting_time[0] = 0;  // First process has 0 waiting time
   for(i = 1; i < n; i++) {
      waiting_time[i] = 0;
      for(j = 0; j < i; j++) {
         waiting_time[i] += burst_time[j];
      }
   }
```

```
    // Calculate turnaround time
    for(i = 0; i < n; i++) {
        turnaround_time[i] = burst_time[i] + waiting_time[i];
    }

    // Calculate average waiting and turnaround times
    for(i = 0; i < n; i++) {
        avg_waiting_time += waiting_time[i];
        avg_turnaround_time += turnaround_time[i];
    }
    avg_waiting_time /= n;
    avg_turnaround_time /= n;

    // Display the process details
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\n", process[i], burst_time[i], waiting_time[i], turnaround_time[i]);
    }
    printf("\nAverage Waiting Time: %.2f", avg_waiting_time);
    printf("\nAverage Turnaround Time: %.2f\n", avg_turnaround_time);
}
int main() {
    fcfs();
    return 0;
}
```

```
--- First Come First Serve (FCFS) Scheduling ---
Enter the number of processes: 4

Enter Burst Time for each process:
P1: 10
P2: 50
P3: 40
P4: 30

Process Burst Time      Waiting Time    Turnaround Time
P1      10              0               10
P2      50              10              60
P3      40              60              100
P4      30              100             130

Average Waiting Time: 42.50
Average Turnaround Time: 75.00
```

2. Shortest Job First (SJF)

```
#include <stdio.h>

void sjf() {
   printf("\n--- Shortest Job First (SJF) Scheduling ---\n");


   int n, i, j, pos, temp;
   printf("Enter the number of processes: ");
   scanf("%d", &n);


   int process[n], burst_time[n], waiting_time[n], turnaround_time[n];
   float avg_waiting_time = 0, avg_turnaround_time = 0;


   printf("\nEnter Burst Time for each process:\n");
   for(i = 0; i < n; i++) {
```

```
    process[i] = i + 1;
    printf("P%d: ", i + 1);
    scanf("%d", &burst_time[i]);
}

// Sort processes by burst time (Selection Sort)
for(i = 0; i < n - 1; i++) {
    pos = i;
    for(j = i + 1; j < n; j++) {
        if(burst_time[j] < burst_time[pos]) {
            pos = j;
        }
    }

    // Swap burst time
    temp = burst_time[i];
    burst_time[i] = burst_time[pos];
    burst_time[pos] = temp;

    // Swap process id
    temp = process[i];
    process[i] = process[pos];
    process[pos] = temp;
}

// Calculate waiting time
waiting_time[0] = 0;  // First process has 0 waiting time
for(i = 1; i < n; i++) {
```

```
        waiting_time[i] = 0;
        for(j = 0; j < i; j++) {
            waiting_time[i] += burst_time[j];
        }
    }
    // Calculate turnaround time
    for(i = 0; i < n; i++) {
        turnaround_time[i] = burst_time[i] + waiting_time[i];
    }

    // Calculate average waiting and turnaround times
    for(i = 0; i < n; i++) {
        avg_waiting_time += waiting_time[i];
        avg_turnaround_time += turnaround_time[i];
    }

    avg_waiting_time /= n;
    avg_turnaround_time /= n;

    // Display the process details
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\n", process[i], burst_time[i], waiting_time[i], turnaround_time[i]);
    }
    printf("\n\nAverage Waiting Time: %.2f", avg_waiting_time);
    printf("\nAverage Turnaround Time: %.2f\n", avg_turnaround_time);
}
int main() {
```

```
    sjf();
    return 0;
}
```

```
--- Shortest Job First (SJF) Scheduling ---
Enter the number of processes: 4

Enter Burst Time for each process:
P1: 20
P2: 40
P3: 10
P4: 50

Process Burst Time      Waiting Time    Turnaround Time
P3      10              0               10
P1      20              10              30
P2      40              30              70
P4      50              70              120


Average Waiting Time: 27.50
Average Turnaround Time: 57.50
```

3. Round Robin Scheduling

```
#include <stdio.h>

void roundRobin() {
    printf("\n--- Round Robin Scheduling ---\n");

    int n, i, j, time, remain, flag = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0;

    printf("Enter the number of processes: ");
```

```
scanf("%d", &n);

remain = n;

int arrival_time[n], burst_time[n], rt[n];

printf("Enter the Time Quantum: ");
scanf("%d", &time_quantum);

printf("\nEnter the Burst Time for each process:\n");
for(i = 0; i < n; i++) {
   printf("P%d: ", i + 1);
   scanf("%d", &burst_time[i]);
   rt[i] = burst_time[i];
   arrival_time[i] = 0;  // All processes arrive at time 0
}

printf("\nProcess\t|Turnaround Time|Waiting Time\n");
// Process until all processes are completed
for(time = 0, i = 0; remain != 0;) {
   if(rt[i] <= time_quantum && rt[i] > 0) {
      time += rt[i];
      rt[i] = 0;
      flag = 1;
   } else if(rt[i] > 0) {
      rt[i] -= time_quantum;
      time += time_quantum;
   }
```

```
        if(rt[i] == 0 && flag == 1) {
            remain--;
            printf("P%d\t|\t%d\t|\t%d\n", i + 1, time - arrival_time[i], time - arrival_time[i] - burst_time[i]);
            wait_time += time - arrival_time[i] - burst_time[i];
            turnaround_time += time - arrival_time[i];
            flag = 0;
        }
        if(i == n - 1)
            i = 0;
        else if(arrival_time[i + 1] <= time)
            i++;
        else
            i = 0;
    }
    printf("\nAverage Waiting Time = %.2f\n", wait_time * 1.0 / n);
    printf("Average Turnaround Time = %.2f\n", turnaround_time * 1.0 / n);
}
int main() {
    roundRobin();
    return 0;
}
```

```
--- Round Robin Scheduling ---
Enter the number of processes: 4
Enter the Time Quantum: 10

Enter the Burst Time for each process:
P1: 50
P2: 30
P3: 60
P4: 20

Process |Turnaround Time|Waiting Time
P4      |       80      |       60
P2      |       100     |       70
P1      |       140     |       90
P3      |       160     |       100

Average Waiting Time = 80.00
Average Turnaround Time = 120.00
```