LAB-2

Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.
→ Priority (pre-emptive & Non-pre-emptive)
→Round Robin (Experiment with different quantum sizes for RR algorithm)

```
#include <stdio.h>

struct Process {
   int id, arrival_time, burst_time, remaining_time, priority;
   int completion_time, turnaround_time, waiting_time;
};

void sortByArrival(struct Process processes[], int n) {
   struct Process temp;
   for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - i - 1; j++) {
         if (processes[j].arrival_time > processes[j + 1].arrival_time) {
            temp = processes[j];
            processes[j] = processes[j + 1];
            processes[j + 1] = temp;
         }
      }
   }
}
void display(struct Process processes[], int n)
{
   printf("\nProcess\tArrival\tBurst\tCompletion\tTAT\tWaiting\n");
   for (int i = 0; i < n; i++) {
      printf("%d\t%d\t%d\t%d\t\t%d\t%d\n", processes[i].id, processes[i].arrival_time,
```

```
            processes[i].burst_time,processes[i].completion_time,
            processes[i].turnaround_time, processes[i].waiting_time);
}
}

void calculateTimes(struct Process processes[], int n) {
    int total_waiting_time = 0, total_turnaround_time = 0;
    for (int i = 0; i < n; i++) {
        processes[i].turnaround_time = processes[i].completion_time - processes[i].arrival_time;
        processes[i].waiting_time = processes[i].turnaround_time - processes[i].burst_time;
        total_waiting_time += processes[i].waiting_time;
        total_turnaround_time += processes[i].turnaround_time;
    }
    display(processes,n);

    printf("Average Waiting Time: %.2f\n", (float)total_waiting_time / n);
    printf("Average Turnaround Time: %.2f\n", (float)total_turnaround_time / n);
}

void priorityPreemptive(struct Process processes[], int n) {
    sortByArrival(processes, n);

    int time = 0, completed = 0, min_priority, selected;

    while (completed < n) {
        min_priority = 9999;
        selected = -1;

        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= time && processes[i].remaining_time > 0 &&
processes[i].priority < min_priority) {
```

```
            min_priority = processes[i].priority;
            selected = i;
        }
    }

    if (selected == -1) {
        time++;
        continue;
    }

    processes[selected].remaining_time--;
    time++;

    if (processes[selected].remaining_time == 0) {
        processes[selected].completion_time = time;
        completed++;
    }
  }


  printf("\nPreemptive Priority Scheduling Results:\n");
  calculateTimes(processes,n);

}


void priorityNonPreemptive(struct Process processes[], int n) {
  sortByArrival(processes, n);

  int time = 0, completed = 0, selected;
```

```
    while (completed < n) {
        selected = -1;
        int min_priority = 9999;

        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= time && processes[i].remaining_time > 0 &&
processes[i].priority < min_priority) {
                min_priority = processes[i].priority;
                selected = i;
            }
        }

        if (selected == -1) {
            time++;
            continue;
        }

        time += processes[selected].burst_time;
        processes[selected].completion_time = time;
        processes[selected].remaining_time = 0;
        completed++;
    }

    printf("\nNon-Preemptive Priority Scheduling Results:\n");
    calculateTimes(processes,n);
}

void roundRobin(struct Process processes[], int n, int quantum) {
```

```
    int time = 0, completed = 0;
    struct Process queue[100];
    int front = 0, rear = 0;

    for (int i = 0; i < n; i++) {
        processes[i].remaining_time = processes[i].burst_time;
    }

    printf("\nRound Robin Scheduling (Quantum: %d):\n", quantum);

    while (completed < n) {
        int found = 0;
        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= time && processes[i].remaining_time > 0) {
                found = 1;

                if (processes[i].remaining_time > quantum) {
                    printf("Process %d executed for %d ms (Time: %d to %d)\n", processes[i].id, quantum,
time, time + quantum);
                    time += quantum;
                    processes[i].remaining_time -= quantum;
                } else {
                    printf("Process %d executed for %d ms and completed (Time: %d to %d)\n",
processes[i].id, processes[i].remaining_time, time, time + processes[i].remaining_time);
                    time += processes[i].remaining_time;
                    processes[i].completion_time = time;
                    processes[i].remaining_time = 0;
                    completed++;
                }
            }
        }
```

```
        if (!found) time++;
    }

    printf("\nRound Robin Results (Quantum: %d):\n", quantum);
    calculateTimes(processes,n);

}

int main() {
    int n, choice, quantum;

    printf("Enter number of processes: ");
    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {
        processes[i].id = i + 1;
        printf("Enter details for Process %d:\n", processes[i].id);
        printf("Arrival Time: ");
        scanf("%d", &processes[i].arrival_time);
        printf("Burst Time: ");
        scanf("%d", &processes[i].burst_time);
        printf("Priority: ");
        scanf("%d", &processes[i].priority);
        processes[i].remaining_time = processes[i].burst_time;

    }
```

```
    printf("\nChoose scheduling algorithm:\n");
    printf("1. Non-Preemptive Priority Scheduling\n");
    printf("2. Preemptive Priority Scheduling\n");
    printf("3. Round Robin Scheduling\n");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            priorityNonPreemptive(processes, n);
            break;
        case 2:
            priorityPreemptive(processes, n);
            break;
        case 3:
            printf("Enter time quantum: ");
            scanf("%d", &quantum);
            roundRobin(processes, n, quantum);
            break;
        default:
            printf("Invalid choice!\n");
    }

    return 0;
}
```

ROUND ROBIN(Q:2)

```
1.round robin(default priority 1)       2.priority(preemptive)  3.priority(non preemptive)
enter the no.of process:5

 enter the details of process 1
arrival time:0
burst time:5
priority1

 enter the details of process 2
arrival time:1
burst time:3
priority1

 enter the details of process 3
arrival time:2
burst time:1
priority1

 enter the details of process 4
arrival time:3
burst time:2
priority1

 enter the details of process 5
arrival time:4
burst time:3
priority1

 enter the choice:1

enter the quantum2
 Round Robin Scheduling(Quantum:2)
Process ID | Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time
         5 |            4 |          3 |              14 |            7 |              10
         4 |            3 |          2 |              11 |            6 |               8
         3 |            2 |          1 |               3 |            0 |               1
         2 |            1 |          3 |              12 |            8 |              11
         1 |            0 |          5 |              13 |            8 |              13
Average Turn around time: 8.600000Average Waiting time: 5.800000
```

ROUND ROBIN(Q:3)

```
1.round robin(default priority 1)       2.priority(preemptive)  3.priority(non preemptive)
enter the no.of process:5

 enter the details of process 1
arrival time:0
burst time:5
priority1

 enter the details of process 2
arrival time:1
burst time:3
priority1

 enter the details of process 3
arrival time:2
burst time:1
priority1

 enter the details of process 4
arrival time:3
burst time:2
priority1

 enter the details of process 5
arrival time:4
burst time:3
priority1

 enter the choice:1

enter the quantum3
 Round Robin Scheduling(Quantum:3)
Process ID | Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time
         1 |            0 |          5 |              14 |            9 |              14
         2 |            1 |          3 |               6 |            2 |               5
         3 |            2 |          1 |               7 |            4 |               5
         4 |            3 |          2 |               9 |            4 |               6
         5 |            4 |          3 |              12 |            5 |               8
Average Turn around time: 7.60
Average Waiting time: 4.80
Process returned 27 (0x1B)   execution time : 18.819 s
```

PRIORITY(NON-PREEMPTIVE)

```
Enter number of processes: 5
Enter details for Process 1:
Arrival Time: 0
Burst Time: 3
Priority: 5
Enter details for Process 2:
Arrival Time: 2
Burst Time: 2
Priority: 3
Enter details for Process 3:
Arrival Time: 3
Burst Time: 5
Priority: 2
Enter details for Process 4:
Arrival Time: 4
Burst Time: 4
Priority: 4
Enter details for Process 5:
Arrival Time: 6
Burst Time: 1
Priority: 1

Choose scheduling algorithm:
1. Non-Preemptive Priority Scheduling
2. Preemptive Priority Scheduling
3. Round Robin Scheduling
1

Non-Preemptive Priority Scheduling Results:

Process Arrival Burst   Completion      TAT     Waiting
1       0       3       3               3       0
2       2       2       11              9       7
3       3       5       8               5       0
4       4       4       15              11      7
5       6       1       9               3       2
Average Waiting Time: 3.20
Average Turnaround Time: 6.20

Process returned 0 (0x0)   execution time : 29.399 s
Press any key to continue.
```

PRIORITY(PREEMPTIVE)

```
Enter number of processes: 5
Enter details for Process 1:
Arrival Time: 0
Burst Time: 3
Priority: 5
Enter details for Process 2:
Arrival Time: 2
Burst Time: 2
Priority: 3
Enter details for Process 3:
Arrival Time: 3
Burst Time: 5
Priority: 2
Enter details for Process 4:
Arrival Time: 4
Burst Time: 4
Priority: 4
Enter details for Process 5:
Arrival Time: 6
Burst Time: 1
Priority: 1

Choose scheduling algorithm:
1. Non-Preemptive Priority Scheduling
2. Preemptive Priority Scheduling
3. Round Robin Scheduling
2

Preemptive Priority Scheduling Results:

Process Arrival Burst   Completion      TAT     Waiting
1       0       3       15              15      12
2       2       2       10              8       6
3       3       5       9               6       1
4       4       4       14              10      6
5       6       1       7               1       0
Average Waiting Time: 5.00
Average Turnaround Time: 8.00

Process returned 0 (0x0)   execution time : 17.419 s
Press any key to continue.
```