WEEK 2

Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.

(a) Priority (pre-emptive & Non-pre-emptive)

(b) Round Robin (Experiment with different quantum sizes for RR algorithm)

1.1.1 Code:

(a) <u>Priority (Non-pre-emptive)</u>

```
#include<stdio.h>

#include<stdlib.h>


struct process {

        int process_id;

        int burst_time;

        int priority;

        int waiting_time;

        int turnaround_time;

};


void find_average_time(struct process[], int);
```

```
void priority_scheduling(struct process[], int);

int main()
{
        int n, i;
        struct process proc[10];

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

for(i = 0; i< n; i++)
        {
        printf("\nEnter the process ID: ");
        scanf("%d", &proc[i].process_id);

        printf("Enter the burst time: ");
        scanf("%d", &proc[i].burst_time);

        printf("Enter the priority: ");
        scanf("%d", &proc[i].priority);
```

```
        }

        priority_scheduling(proc, n);
        return 0;
}

void find_waiting_time(struct process proc[], int n, int wt[])
{
        int i;
        wt[0] = 0;

        for(i = 1; i< n; i++)
        {
        wt[i] = proc[i - 1].burst_time + wt[i - 1];
        }
}

void find_turnaround_time(struct process proc[], int n, int wt[], int tat[])
{
        int i;
        for(i = 0; i< n; i++)
```

```
        {
        tat[i] = proc[i].burst_time + wt[i];
        }
}

void find_average_time(struct process proc[], int n)
{
        int wt[10], tat[10], total_wt = 0, total_tat = 0, i;

        find_waiting_time(proc, n, wt);
        find_turnaround_time(proc, n, wt, tat);

        printf("\nProcess ID\tBurst Time\tPriority\tWaiting Time\tTurnaround Time");

        for(i = 0; i< n; i++)
        {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", proc[i].process_id, proc[i].burst_time, proc[i].priority, wt[i],              tat[i]);
        }
```

```
        printf("\n\nAverage Waiting Time = %f", (float)total_wt/n);
        printf("\nAverage Turnaround Time = %f\n", (float)total_tat/n);
}
void priority_scheduling(struct process proc[], int n)
{
        int i, j, pos;
        struct process temp;
        for(i = 0; i< n; i++)
        {
        pos = i;
        for(j = i + 1; j < n; j++)
        {
          if(proc[j].priority< proc[pos].priority)
        pos = j;
        }
        temp = proc[i];
        proc[i] = proc[pos];
        proc[pos] = temp;
        }
        find_average_time(proc, n);
}
```

(b) <u>Round Robin (Non-pre-emptive)</u>

```
#include <stdio.h>
#include <stdbool.h>

int turnarroundtime(int processes[], int n, int bt[], int wt[], int tat[]) {
   for (int i = 0; i < n ; i++)
   tat[i] = bt[i] + wt[i];
   return 1;
}

int waitingtime(int processes[], int n, int bt[], int wt[], int quantum)
{
   int rem_bt[n];
   for (int i = 0 ; i < n ; i++)
   rem_bt[i] = bt[i];
   int t = 0;

         while (1)
         {
         bool done = true;
```

```
for (int i = 0 ; i < n; i++)
{
if (rem_bt[i] > 0)
{
        done = false;
        if (rem_bt[i] > quantum)
        {
        t += quantum;
        rem_bt[i] -= quantum;
        }

        else
        {
        t = t + rem_bt[i];
        wt[i] = t - bt[i];
        rem_bt[i] = 0;
        }
}
}
if (done == true)
```

```
        break;
    }
    return 1;
}

int findavgTime(int processes[], int n, int bt[], int quantum) {
        int wt[n], tat[n], total_wt = 0, total_tat = 0;

        waitingtime(processes, n, bt, wt, quantum);
        turnarroundtime(processes, n, bt, wt, tat);

        printf("\n\nProcesses\t\t Burst Time\t\t Waiting Time\t\t turnaround time\n");
        for (int i=0; i<n; i++)
        {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
       printf("\n\t%d\t\t\t%d\t\t\t%d\t\t\t%d\n",i+1, bt[i], wt[i], tat[i]);
        }

        printf("\nAverage waiting time = %f", (float)total_wt / (float)n);
        printf("\nAverage turnaround time = %f", (float)total_tat / (float)n);
```

```
        return 1;
}

int main()
{
  int n, processes[n], burst_time[n], quantum;
  printf("Enter the Number of Processes: ");
  scanf("%d",&n);

  printf("\nEnter the quantum time: ");
  scanf("%d",&quantum);

        int i=0;
        for(i=0;i<n;i++)
        {
        printf("\nEnter the process: ");
      scanf("%d",&processes[i]);
        printf("Enter the Burst Time:");
      scanf("%d",&burst_time[i]);
  }
```

```
    findavgTime(processes, n, burst_time, quantum);
    return 0;
}
```

OUTPUT

```
Enter the number of processes: 3

Enter the process ID: 1
Enter the burst time: 10
Enter the priority: 3

Enter the process ID: 2
Enter the burst time: 8
Enter the priority: 2

Enter the process ID: 3
Enter the burst time: 5
Enter the priority: 1

Process ID      Burst Time      Priority        Waiting Time    Turnaround Time
3               5               1               0               5
2               8               2               5               13
1               10              3               13              23

Average Waiting Time = 6.000000
Average Turnaround Time = 13.666667
```

```
Enter the Number of Processes: 3

Enter the quantum time: 2

Enter the process: 1
Enter the Burst Time:4

Enter the process: 2
Enter the Burst Time:3

Enter the process: 3
Enter the Burst Time:5


Processes                Burst Time              Waiting Time            turnaround time

        1                       4                       4                       8

        2                       3                       6                       9

        3                       5                       7                       12

Average waiting time = 5.666667
Average turnaround time = 9.666667
```

Lab 03

write a c prog to similar the following CPU scheduling algo to find the turnaround time & waiting time.

→ Priority [Non - pre emptive]

→ Round Robin

① #include <stdio.h>

#include <conio.h>

int main()

{

int n, i, bt[20], priority[20], at[20], j, temp, wt[20], tat[20], sum 20, conv 20, avgtat = 0;

printf("Enter no of processes");

scanf("%d", &n);

for (i=0; i<n; i++)

{ printf("Enter AT for P[%d]:");

scanf("%d", &a+[i]);

printf("Enter burst time for P[%d]:");

scanf("%d", &bt[i]);

printf("Enter burst time for P[%d]:");

scanf("%d", priority[i]);

}

for (i=0; i<n-1; i++)

for (j=0; j<n-1-i; j++)

{ if (priority[j] > priority[j+1])

{ temp = priority[j];

priority[j+1] = temp;

temp = bt[j];

bt[j+1] = temp;

temp = at[

at[j] =

at[j+1]

}

}

for (i=0;

printf("%d

avgwt+

Sum+

}

float avg

float avgt

printf("

printf("Total av

getche()

return 0;

}

output

Enter no

enter no

enter burst

enter priorit

enter arriva

enter burs

enter pri

enter ar

enter b

enter a

enter

enter

```
        temp = at[j];
        at[j] = at[j+1];
        at[j+1] = temp;
      }
    }
  }
  for (i=0; i<n; i++)
    printf("%d\n", wt[i]);
    avgwt += wt[i];
    sumt = bt[i];
  }
  float avgwt = (float) avgwt/n;
  float avgtat = (float) avgtat/n;
  printf("\n total avg waiting time: %f", avgwt);
printf("Total avg turnaround time: %f", avgtat);
  getche();
  return 0;
}
```

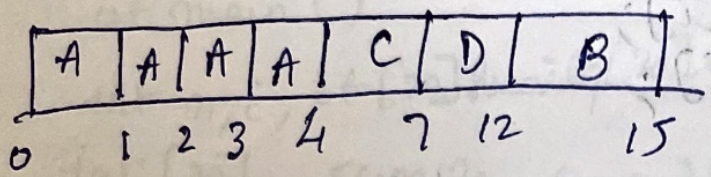
output

```
Enter no of process = 4
enter no of time for P(1) : 0
enter burst time for P(1): 4
enter priority of P(1) : 3
enter arrival time for P(2) : 1
enter burst time for P(2): 3
enter priority of P(2): 4
```

Total avg waiting time : 4.25

Total avg turnaround time : 8.0

Process	AT	CPU - T	Priority
A	0	4	3
B	1	3	4
C	2	3	6
D	3	5	5



A(4) - A(3) A(2) A(1) B(5) B(4) B(3)

B(4) B(4) C(6) D(5)

C(6) D(5)

TAT = 4, 14, 5, 9

ATAT = 8

WT = 0, 11, 2, 4

AWT = $\frac{17}{4}$ = 4.25