LABWORK – 6

Objective: Write C programs which implement the following CPU scheduling algorithms.

- Round Robin
- Priority Scheduling

Description:

- **Round Robin**

Each process is assigned a time interval called its quantum(time slice) If the process is still running at the end of the quantum the CPU is preempted and given to another process, and this continues in circular fashion, till all the processes are completely executed.

Algorithm:

Step 1: Initialize all the structure elements

Step 2: Receive inputs from the user to fill process id,burst time and arrival time.

Step 3: Calculate the waiting time for all the process id.

i) The waiting time for first instance of a process is calculated as:

a[i].waittime=count + a[i].arrivt

ii) The waiting time for the rest of the instances of the process is calculated as:

a) If the time quantum is greater than the remaining burst time then waiting time is calculated as:

a[i].waittime=count + tq

b) Else if the time quantum is greater than the remaining burst time then waiting time is calculated as:

a[i].waittime=count - remaining burst time

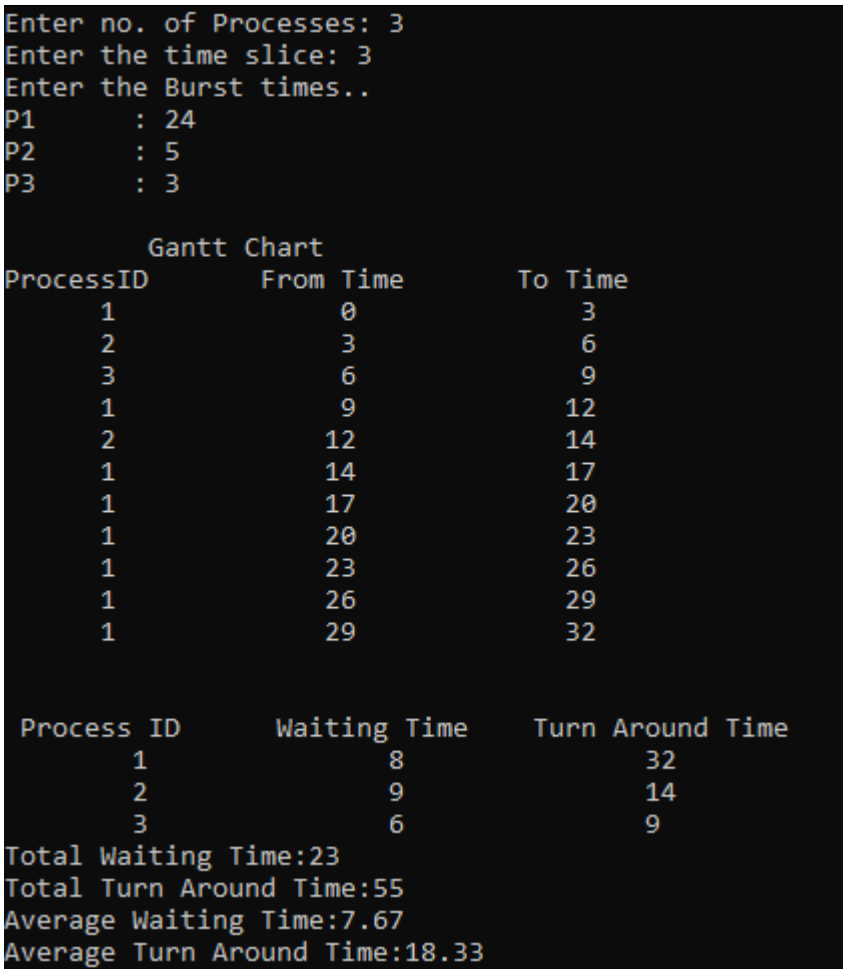
Step 4: Calculate the average waiting time and average turnaround time

Step 5: Print the results of the step 4.

Sample Code of Round Robin Algorithm:

```
#include<stdio.h>

void main(){
        int i,tbt=0,nop,ts=0,flag[20], rem[20];
        int from,wt[20],tt[20],b[20], twt=0,ttt=0;
        int dur;
        float awt,att;
        printf("Enter no. of Processes: ");
        scanf("%d",&nop);
        printf("Enter the time slice: ");
        scanf("%d",&ts);
```

```
        printf("Enter the Burst times..\n");
        for(i=0;i<nop;i++){
                wt[i]=tt[i]=0;
                printf("P%d\t: ",i+1);
                scanf("%d",&b[i]);
                rem[i]=b[i];
                tbt+=b[i];
                flag[i]=0;
                }

        from=0;
        i=0;
        printf("\n\t Gantt Chart");
        printf("\nProcessID\tFrom Time\tTo Time\n");
        while(from<tbt){
                if(!flag[i]){
                        if(rem[i]<=ts){
                                dur=rem[i];
                                flag[i]=1;
                                tt[i]=dur+from;
                                wt[i]=tt[i]-b[i];
                                }
                        else
                                dur=ts;
                printf("%7d%15d%15d\n",i+1, from,from+dur);
                rem[i] -= dur;
                from += dur;
                }
        i=(i+1)%nop;}
        for(i=0;i<nop;i++){
                twt+=wt[i];
                ttt+=tt[i];
                }

printf("\n\n Process ID \t Waiting Time \t Turn Around Time");
for(i=0;i<nop;i++){
printf("\n\t%d\t\t%d\t\t%d",i+1,wt[i],tt[i]);
}

awt=(float)twt/(float)nop;
att=(float)ttt/(float)nop;
printf("\nTotal Waiting Time:%d",twt);
printf("\nTotal Turn Around Time:%d",ttt);
printf("\nAverage Waiting Time:%.2f",awt);
printf("\nAverage Turn Around Time:%.2f\n",att);
}
```



- **Priority Scheduling**

Each process is assigned a priority and executable process with highest priority is allowed to run

Algorithm:

Step1:Get the number of process,burst time and priority.

Step2:Using for loopi=0 to n-1 do step 1 to 6.

Step3:If i=0,wait time=0,T[0]=b[0];

Step4:T[i]=T[i-1]+b[i] and wt[i]=T[i]-b[i].

Step5: Total waiting time is calculated by adding the waiting time for lack process.

Step6: Total turn around time is calculated by adding all total time of each process.

Step7: Calculate Average waiting time by dividing the total waiting time by total numberof process.

Step8: Calculate Average turn around time by dividing the total time by the number of

process.

Step9: Display the result.

TASK: Write a program which implement the **Priority scheduling algorithm** according to output below.

OUTPUT: