1) AIM: CPU scheduling algorithms to find turnaround time and waiting time

a) Round Robin **b) Priority**

a) ROUND ROBIN

DESCRIPTION: To aim is to calculate the average waiting time. There will be a time slice, each process should be executed within that time-slice and if not it will go to the waiting state so first check whether the burst time is less than the time-slice. If it is less than it assign the waiting time to the sum of the total times. If it is greater than the burst-time then subtract the time slot from the actual burst time and increment it by time-slot and the loop continues until all the processes are completed.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where

No. of time slice for process (n) = burst time process (n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

a) Waiting time for process (n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU from process(n-1)

b) Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7: Calculate

c) Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number ofprocess

Step 8: Stop the process

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
```

```
#include<string.h>
void main()
{
char p[10][5];
int et[10],wt[10],timer=3,count,pt[10],rt,i,j,totwt=0,t,n=5,found=0,m;
float avgwt;
clrscr();
for(i=0;i<n;i++)
{
printf("enter the process name : ");
scanf("%s",&p[i]);
printf("enter the processing time : ");
scanf("%d",&pt[i]);
}
m=n;
wt[0]=0;
i=0;
do
{
if(pt[i]>timer)
{
rt=pt[i]-timer;
strcpy(p[n],p[i])
;pt[n]=rt;
et[i]=timer;
n++;
}
else
{
et[i]=pt[i];
}
```

```
i++;
wt[i]=wt[i-1]+et[i-1];
}while(i<n);
count=0;
for(i=0;i<m;i++)
{
for(j=i+1;j<=n;j++)
{
if(strcmp(p[i],p[j])==0)
{
count++;
found=j;
}}
if(found!=0)
{
wt[i]=wt[found]-(count*timer);
count=0;
found=0;
}
}
for(i=0;i<m;i++)
{
totwt+=wt[i];
}
avgwt=(float)totwt/m;
for(i=0;i<m;i++)
{
printf("\n%s\t%d\t%d",p[i],pt[i],wt[i]);
}
printf("\ntotal waiting time %d\n",totwt);
printf("total avgtime %f",avgwt);
```

}

OUTPUT:

Enter the process name:a

Enter the processing time:4

Enter the process name:b

Enter the processing time:3

Enter the process name:c

Enter the processing time:2

Enter the process name:d

Enter the processing time:5

Enter the process name:e

Enter the processing time:1

p_name	p_time	w_time
a	4	9
b	3	3
c	2	6
d	5	10
e	1	11

B) PRIORITY

DESCRIPTION: To calculate the average waiting time in the priority algorithm, sort the burst times according to their priorities and then calculate the average waiting time of the processes. The waiting time of each process is obtained by summing up the burst times of all the previous processes.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Sort the ready queue according to the priority number.

Step 5: Set the waiting of the first process as _0‘ and its burst time as its turnaround time

Step 6: Arrange the processes based on process priority

Step 7: For each process in the Ready Q calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 8: Calculate

c) Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number of process Print the results in an order.

Step 9: Stop

PROGRAM:

```
#include<stdio.h>
#include<conio.h
>void main()
{
int i,j,*p,*at,*bt,*wt,*tat,*ct,n,temp,temp1,temp2;int
sum=0;
float avg
clrscr();
printf("enter num of process:\n");
scanf("%d",&n);
p=(int *)calloc(n,sizeof(int));
at=(int *)calloc(n,sizeof(int));
bt=(int *)calloc(n,sizeof(int));
printf("enter priorities:\n");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
printf("enter arrival time:\n");
for(i=0;i<n;i++)
scanf("%d",&at[i]); printf("enter
burst times:\n");for(i=0;i<n;i++)
```

```
scanf("%d",&bt[i]);
for(i=0;i<n;i++)
{
for(j=0;j<n-i-1;j++)
{                       if(p[j+1]<p[j])
{
temp=p[j]; p[j]=p[j+1];
p[j+1]=temp; temp1=at[j];
at[j]=at[j+1]; at[j+1]=temp1;
temp2=bt[j]; bt[j]=bt[j+1];
bt[j+1]=temp2;
}
}
}
for(i=0;i<n;i++)
printf("%d\t\t%d\t\t%d\n",p[i],at[i],bt[i]);
wt[0]=0;
tat[0]=wt[0]+bt[0];
//ct[0]=at[0]+tat[0];
for(i=1;i<n;i++){
wt[i]=bt[i-1]+wt[i-1];
tat[i]=wt[i]+bt[i];
}
for(i=0;i<n;i++)
printf("%d\t\t%d\n",wt[i],tat[i]);
printf("priorities\tarrival time\tburst time\twaiting time\tturn around time");
for(i=0;i<n;i++){
printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",p[i],at[i],bt[i],wt[i],tat[i]);
}
for(i=0;i<n;i++)
printf("%d",wt[i]);
//            printf("Waiting time of p2 is %d",wt[1]);

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

```
sum=sum+wt[0]+wt[1]+wt[2];
avg=sum/n;
printf("average time is %f and wating time is %d", avg,sum);
getch();
}
```

OUTPUT:

```
Enter num of process : 3
Enter Priorities:3 1  2
Enter Arrival Times : 0 1 2
Enter Burst Time:      1 2  3
```

Priorities	Arrival Time	Burst Time	Waiting Time	TurnAround Time
1	1	2	0	2
2	2	3	2	5
3	0	1	5	6

```
Average Time is 2.0
Waiting time          is 7
```