ITE2002-OPERATING SYSTEM

LAB

WINTER SEM 20-21

Assessment – 2

CAT-1

Name :Pravin G

Reg No :19BIT0393

Slot :L41-42

Algorithms:

read bt at,no of process.
If bt>5 push in fcfs
Else sjf

Compute sjf in queue 2
Compute fcfs in queue1
Display values

Code:

```
#include<stdio.h>
int n;
int fcfsq[20],sjfq[20],n1,n2;
struct process
{
    int pid;
    int at;
    int bt;
    int wt;
    int tt;
    int rbt;
    int status;
}P[20];



void executefcfs()
{
    int curpro,cur,pre,curtime=0;
    int totwt=0,tottt=0;
    int i,idx;
    printf("\nOrder of Execution : ");
    for(i=0;i<n1;i++)
    {
```

```
        cur=fcfsq[i];
        if(P[cur].at>curtime)
            curtime=P[cur].at;
        printf(" P%d",cur+1);
        curtime+=P[cur].bt;
        P[cur].tt=curtime-P[cur].at;
        P[cur].wt=P[cur].tt-P[cur].bt;
        totwt+=P[cur].wt;
        tottt+=P[cur].tt;
    }
    printf("\n\nProcess    AT\tBt\tWT\tTT\n");
    for(i=0;i<n2;i++)
    {
        idx=fcfsq[i];
        printf("P-%d--
>  %d\t%d\t%d\t%d\n",idx+1,P[idx].at,P[idx].bt,P[idx].wt,P[i
dx].tt);
    }
    printf("\nAvarage   waiting  time : %0.2f",(totwt/(float
)n));
    printf("\nAvarage turnaround time : %0.2f",(tottt/(float
)n));

}
int isNotDone()
{
    int i,idx;
    for(i=0;i<n2;i++)
    {
        idx=sjfq[i];
        if(P[idx].status!=1)
            return 1;
    }
    return 0;
}
int minpro()
{
    int min=1000,minidx=-1,i,idx;
```

```
    for(i=0;i<n2;i++)
    {
        idx=sjfq[i];
        if(P[idx].rbt<=min && P[idx].status==0)
        {
            if(P[idx].rbt<min || P[idx].at<P[minidx].at)
            {
            minidx=idx;
            min=P[idx].rbt;
            }
        }
    }
    return minidx;
}
void executesjf()
{
    int curpro,cur,pre,curtime=0;
    int totwt=0,tottt=0;
    int idx,i;
    printf("\nOrder Of Exection : ");
    while(isNotDone())
    {
        for(i=0;i<n2;i++)
        {
            idx=sjfq[i];
            if(P[idx].status==-1 && P[idx].at<=curtime)
            {
                P[idx].status=0;
            }
        }
        cur=minpro();
        if(cur==-1)
        {
            curtime+=1;
            continue;
        }
        if(cur!=pre)
            printf(" P%d",cur+1);
```

```
        P[cur].rbt-=1;
        curtime+=1;
        if(P[cur].rbt==0)
        {
            P[cur].tt=curtime-P[cur].at;
            P[cur].wt=P[cur].tt-P[cur].bt;
            tottt+=P[cur].tt;
            totwt+=P[cur].wt;
            P[cur].status=1;
        }
        pre=cur;
    }
    printf("\nProcess AT\tBT\tWT\tTT\n");
    for(i=0;i<n2;i++)
    {
        idx=sjfq[i];
        printf("P-%d--
>  %d\t%d\t%d\t%d\n",idx+1,P[idx].at,P[idx].bt,P[idx].wt,P[i
dx].tt);
    }
    printf("\nAvarage   waiting  time : %0.2f",(totwt/(float
)n));
    printf("\nAvarage turnaround time : %0.2f",(tottt/(float
)n));
}
void order()
{
    int i,j;
    struct process temp;
    printf("\nProcess Arraival Order ");
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(P[i].at > P[j].at)
            {
                temp = P[i];
                P[i] = P[j];
```

```
                P[j] = temp;
            }
        }
    printf(" P%d",P[i].pid);
    }

}
int main()
{
    printf("CAT1:-\n\n");

    int i;

    printf("Enter no of process : ");
    scanf("%d",&n);
    printf("Enter Arraival times of Each Process :-\n");
    for(i=0;i<n;i++)
    {
        printf("P%d --> ",i+1);
        scanf("%d",&P[i].at);
        P[i].pid=i+1;
        P[i].wt=0;
        P[i].tt=0;
        P[i].status=-1;
    }
    printf("Enter Burst times of Each Process :-\n");
    for(i=0;i<n;i++)
    {
        printf("P%d --> ",i+1);
        scanf("%d",&P[i].bt);
        P[i].rbt=P[i].bt;
    }

    order();

    for(i=0;i<n;i++)
    {
        if(P[i].bt>5)
```

```
            fcfsq[n1++]=i;
        else
            sjfq[n2++]=i;
    }
    executefcfs();
    executesjf();

}
```

Output:

```
CAT1:-

Enter no of process : 10
Enter Arraival times of Each Process :-
P1 --> 0
P2 --> 2
P3 --> 1
P4 --> 1
P5 --> 3
P6 --> 00
P7 --> 0
P8 --> 6
P9 --> 4
P10 --> 0
Enter Burst times of Each Process :-
P1 --> 8
P2 --> 5
P3 --> 6
P4 --> 3
P5 --> 9
P6 --> 4
P7 --> 2
P8 --> 7
P9 --> 15
P10 --> 1
```

```
Process Arraival Order  P1 P6 P7 P10 P4 P3 P2 P5 P9 P8
Order of Execution :  P1 P6 P8 P9 P10

Process   AT    Bt      WT      TT
P-1-->  0       8       0       8
P-6-->  1       6       7       13
P-8-->  3       9       11      20
P-9-->  4       15      19      34
P-10-->  6      7       32      39

Avarage   waiting  time : 6.90
Avarage turnaround time : 11.40
Order Of Exection :  P4 P3 P5 P2 P7
Process AT      BT      WT      TT
P-2-->  0       4       6       10
P-3-->  0       2       1       3
P-4-->  0       1       0       1
P-5-->  1       3       2       5
P-7-->  2       5       8       13

Avarage   waiting  time : 1.70
Avarage turnaround time : 3.20
```