NAME OF THE EXPERIMENT:CPU Scheduling Techniques SJF, Round Robin

AIM:Write C Programs to simulate the following CPU scheduling algorithms:

a)SJF b) Round Robin

a) SJF 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: Start the Ready Q according the shortest Burst time by sorting

according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as ‘0’ and its turnaround time as its

burst time.

Step 6: For each process in the ready queue, calculate

Waiting time for process(n)= waiting time of process (n-1)+Burst time of process(n-

1)

Turn around time for Process(n)= waiting time of Process(n)+ Burst time for

process(n)

Step 7: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

SOURCE CODE:

#include<stdio.h>;

void main(){

int pid[10],bt[10],wt[10],tat[10],n,twt=0,ttat=0,i,j,t;

float awt,atat;

printf(“Enter no.of processes:”);

scanf(“%d”,&n);

printf(“\n Enter burst times:”);

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

scanf(“%d”,&bt[i]);

printf(“\n Enter PID:”);

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

scanf(“%d”,& pid[i]);

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

{

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

{

if(bt[i]>bt[j])

{

t=bt[i];

bt[i]=bt[j];

bt[j]=t;

t=pid[i];

pid[i]=pid[j];

pid[j]=t;

}}}

wt[0]=0;

tat[0]=bt[0];

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

{

wt[i]=tat[i-1];

tat[i]=bt[i]+wt[i];

}

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

{

ttat= ttat+tat[i];

twt=twt+wt[i];

}

printf(“\n PID \t BT \t WT \t TAT”);

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

printf(”\n %d\t%d\t%d\t%d”,pid[i],bt[i],wt[i],tat[i]);

awt=(float)twt/n;

atat=(float)ttat/n;

printf(“\nAvg. Waiting Time=%f”,awt);

printf(“\nAvg. Turn around time=%f” ,atat);

}