Name-Rishabh Saxena

Roll no-28

Question allotted --5

#include<stdio.h>

#include<conio.h>

void main()

{

int ts,pid[10],need[10],wt[10],tat[10],i,j,n,n1;

int bt[10],flag[10],ttat=0,twt=0;

float awt,atat;

printf("\t\t ROUND ROBIN SCHEDULING");

printf("\nEnter the number of Processors \n");

scanf("%d",&n);

n1=n;

printf("\n Enter the Timeslice \n");

scanf("%d",&ts);

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

{

printf("\n Enter the process ID %d",i);

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

printf("\n Enter the Burst Time for the process");

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

need[i]=bt[i];

}

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

{

flag[i]=1;

wt[i]=0;

}

while(n!=0)

{

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

{

if(need[i]>=ts)

{

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

{

if((i!=j)&&(flag[i]==1)&&(need[j]!=0))

wt[j]+=ts;

}

need[i]-=ts;

if(need[i]==0)

{

flag[i]=0;

n--;

}

}

else

{

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

{

if((i!=j)&&(flag[i]==1)&&(need[j]!=0))

wt[j]+=need[i];

}

need[i]=0;

n--;

flag[i]=0;

}

}

}

for(i=1;i<=n1;i++)

{

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

twt=twt+wt[i];

ttat=ttat+tat[i];

}

awt=(float)twt/n1;

atat=(float)ttat/n1;

printf("\n\n ROUND ROBIN SCHEDULING ALGORITHM \n\n");

printf("\n\n Process \t Process ID \t BurstTime \t Waiting Time \t TurnaroundTime \n ");

for(i=1;i<=n1;i++)

{

printf("\n %5d \t %5d \t\t %5d \t\t %5d \t\t %5d \n", i,pid[i],bt[i],wt[i],tat[i]);

}

printf("\n The average Waiting Time=%.2f",awt);

printf("\n The average Turn around Time=%.2f",atat);

getch();

}

1- Create an array **rem\_bt[]** to keep track of remaining

burst time of processes. This array is initially a

copy of bt[] (burst times array)

2- Create another array **wt[]** to store waiting times

of processes. Initialize this array as 0.

3- Initialize time : t = 0

4- Keep traversing the all processes while all processes

are not done. Do following for i'th process if it is

not done yet.

a- If rem\_bt[i] > quantum

(i) t = t + quantum

(ii) bt\_rem[i] -= quantum;

c- Else // Last cycle for this process

(i) t = t + bt\_rem[i];

(ii) wt[i] = t - bt[i]

(ii) bt\_rem[i] = 0; // This process is over