b) ROUND ROBIN ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue &amp; time quantum or

time slice

Step 3: For each process in the ready Q, assign the process id &amp;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

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)

Turn around 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

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 ts,bt1[10],wt[10],tat[10],i,j=0,n,bt[10],ttat=0,twt=0,tot=0;

float awt,atat;

printf(“Enter the number of Processes \n”);

scanf(“%d”,&n);

printf(“\n Enter the Timeslice \n”);

scanf(”%d”,&ts);

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

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

scanf(“%d”,bt1[i]);

bt[i]=bt1[i];

}

while(j<n){

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

if(bt[i]>0){

if(bt[i]>=ts){

tot+=ts;

bt[i]=bt[i]-ts;

if(bt[i]==0){

j++;

tat[i]=tot;

}}

else{

tot+=bt[i];

bt[i]=0;

j++;

tat[i]=tot;

}}}}

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

wt[i]=tat[i]-bt1[i];

twt=twt+wt[i];

ttat=ttat+tat[i];

}

awt=(float)twt/n;

atat=(float)ttat/n;

printf(“\n PID \t BT \t WT \t TAT\n”);

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

printf(“\n %d \t %d \t %d \t %d \t\n”,i,bt1[i],wt[i],tat[i]);

}

printf(“\n The average Waiting Time=%f”,awt);

printf(“\n The average Turn around Time=%f” ,atat);

}