OPERATING SYSTEMS

Questions:

1. Write a C program to implement the following CPU Scheduling Algorithms.

(i) FCFS Scheduling.

(ii) SJF and SRTF Scheduling.

(iii) Priority Scheduling .(Premptive and Non Premptive)

(iv) Round Robin Scheduling.

2. Write a C program to implement Bankers Algorithm For Deadlock Avoidance.

1.

i) FCFS

CODE:

#include<stdio.h>

#include<stdlib.h>

typedef struct process{

int at,bt,ct,tat,wt,id;

}process;

process Atthemoment[20];

process hastocome[20];

process completed[20];

void swap(process \*a,process \*b){

process c=\*a;

\*a=\*b;

\*b=c;

}

int idnotincomp(int id,int d){

int c=0,i,j;

for(i=0;i<d;i++){

if(completed[i].id==id){

c++;

}

}

if(c==0){

return 1;

}

else{

return 0;

}

}

void initatthemoment(){

int i;

for(i=0;i<20;i++){

Atthemoment[i].at='\0';

Atthemoment[i].bt='\0';

Atthemoment[i].ct='\0';

Atthemoment[i].tat='\0';

Atthemoment[i].wt='\0';

}

}

void inithastocome(){

int i;

for(i=0;i<20;i++){

hastocome[i].at='\0';

hastocome[i].bt='\0';

hastocome[i].ct='\0';

hastocome[i].tat='\0';

hastocome[i].wt='\0';

}

}

process shortestarrival(int z){

int i,j;

process shor=hastocome[0];

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

if(hastocome[i].at<shor.at){

shor=hastocome[i];

}

}

return shor;

}

process shortest(int m){

int i;

process shor=Atthemoment[0];

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

if(Atthemoment[i].at<shor.at){

shor=Atthemoment[i];

}

}

return shor;

}

int main(){

process p[20];

process shor;

int i,j,k,n,m=0,d=0,z,samear=0;

float awt=0,atat=0;

printf("Enter the number of processes:");

scanf("%d",&n);

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

p[i].id=i+1;

printf("Enter the arrival time for process[%d]:",i+1);

scanf("%d",&p[i].at);

printf("Enter the burst time for process[%d]:",i+1);

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

}

for(i=0;i<n-1;++i){

for(j=0;j<n-i-1;++j){

if(p[j].at>p[j+1].at){

swap(&p[j],&p[j+1]);

}

}

}

Atthemoment[m++]=p[0];

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

samear=0;

if(p[0].at==p[i].at){

samear++;

Atthemoment[m++]=p[i];

}

}

if(samear>0){

shor=shortest(m);

shor.wt=0;

shor.ct=shor.bt+shor.at;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;}

while(d<n){

m=0;

z=0;

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

if(idnotincomp(p[i].id,d)==1){

if(p[i].at<=completed[d-1].ct){

Atthemoment[m++]=p[i];

}

}

}

if(Atthemoment[0].bt=='\0'){

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

if(idnotincomp(p[i].id,d)==1){

hastocome[z++]=p[i];

}

}

shor=shortestarrival(z);

shor.wt=0;

shor.ct=shor.at+shor.bt;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

inithastocome();

initatthemoment();

}

else{

shor=shortest(m);

shor.wt=completed[d-1].ct-shor.at;

shor.ct=completed[d-1].ct+shor.bt;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

inithastocome();

initatthemoment();

}

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nProcess\t Arrival\t Burst\t Waiting\t Turn-around");

for(i=0;i<d;i++){

printf("\n p%d\t %d\t\t %d\t %d\t\t\t%d\n",completed[i].id,completed[i].at,completed[i].bt,completed[i].wt,completed[i].tat);

awt=awt+completed[i].wt;

atat=atat+completed[i].tat;

}

awt=awt/n;

atat=atat/n;

printf("\nAverage waiting time: %.3f\n",awt);

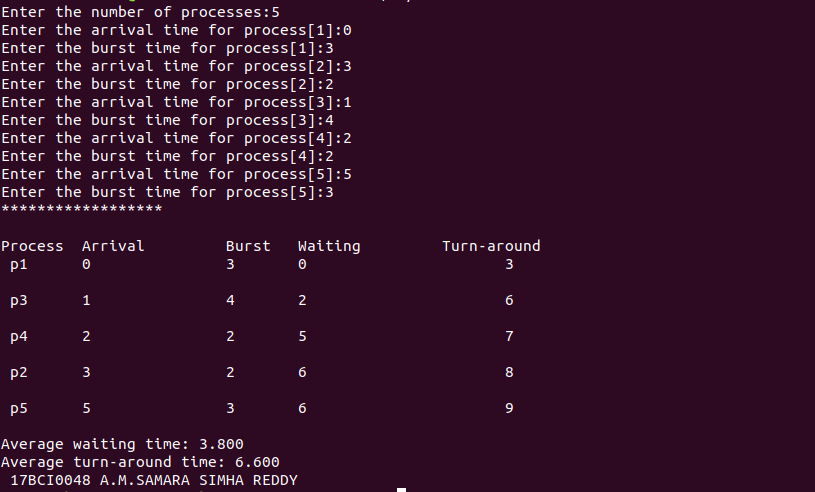
printf("Average turn-around time: %.3f\n",atat);

printf(" 17BCI0048 A.M.SAMARA SIMHA REDDY\n");

return 0;

}

output:



1.

ii) SJF

code:

#include<stdio.h>

#include<stdlib.h>

typedef struct process{

int at,bt,ct,tat,wt,id;

}process;

process Atthemoment[20];

process hastocome[20];

process completed[20];

process more[20];

process shortburst(process more[],int k){

int i,j;

process shor=more[0];

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

if(more[i].bt<shor.bt){

shor=more[i];

}

}

return shor;

}

void swap(process \*a,process \*b){

process c=\*a;

\*a=\*b;

\*b=c;

}

int idnotincomp(int id,int d){

int c=0,i,j;

for(i=0;i<d;i++){

if(completed[i].id==id){

c++;

}

}

if(c==0){

return 1;

}

else{

return 0;

}

}

void initatthemoment(){

int i;

for(i=0;i<20;i++){

Atthemoment[i].at='\0';

Atthemoment[i].bt='\0';

Atthemoment[i].ct='\0';

Atthemoment[i].tat='\0';

Atthemoment[i].wt='\0';

}

}

void inithastocome(){

int i;

for(i=0;i<20;i++){

hastocome[i].at='\0';

hastocome[i].bt='\0';

hastocome[i].ct='\0';

hastocome[i].tat='\0';

hastocome[i].wt='\0';

}

}

process shortestarrival(int z){

int i,j,c=0;

process shor=hastocome[0];

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

if(hastocome[i].at<shor.at){

shor=hastocome[i];

}

}

for(i=0;i<z;i++){

if(shor.at==hastocome[i].at){

more[c++]=hastocome[i];

}

}

shor=shortburst(more,c);

return shor;

}

process shortest(int m){

int i;

process shor=Atthemoment[0];

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

if(Atthemoment[i].bt<shor.bt){

shor=Atthemoment[i];

}

}

return shor;

}

int main(){

process p[20];

process shor;

int i,j,k,n,m=0,d=0,z,samear=0;

float awt=0,atat=0;

printf("Enter the number of processes:");

scanf("%d",&n);

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

p[i].id=i+1;

printf("Enter the arrival time for process[%d]:",i+1);

scanf("%d",&p[i].at);

printf("Enter the burst time for process[%d]:",i+1);

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

}

for(i=0;i<n-1;++i){

for(j=0;j<n-i-1;++j){

if(p[j].at>p[j+1].at){

swap(&p[j],&p[j+1]);

}

}

}

Atthemoment[m++]=p[0];

samear=0;

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

if(p[0].at==p[i].at){

samear++;

Atthemoment[m++]=p[i];

}

}

shor=shortest(m);

shor.wt=0;

shor.ct=shor.bt+shor.at;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

while(d<n){

m=0;

z=0;

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

if(idnotincomp(p[i].id,d)==1){

if(p[i].at<=completed[d-1].ct){

Atthemoment[m++]=p[i];

}

}

}

if(Atthemoment[0].bt=='\0'){

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

if(idnotincomp(p[i].id,d)==1){

hastocome[z++]=p[i];

}

}

shor=shortestarrival(z);

shor.wt=0;

shor.ct=shor.at+shor.bt;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

inithastocome();

initatthemoment();

}

else{

shor=shortest(m);

shor.wt=completed[d-1].ct-shor.at;

shor.ct=completed[d-1].ct+shor.bt;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

inithastocome();

initatthemoment();

}

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nProcess\t Arrival\t Burst\t Waiting\t Turn-around");

for(i=0;i<d;i++){

printf("\n p%d\t %d\t\t %d\t %d\t\t\t%d\n",completed[i].id,completed[i].at,completed[i].bt,completed[i].wt,completed[i].tat);

awt=awt+completed[i].wt;

atat=atat+completed[i].tat;

}

awt=awt/n;

atat=atat/n;

printf("Average waiting time: %.3f\n",awt);

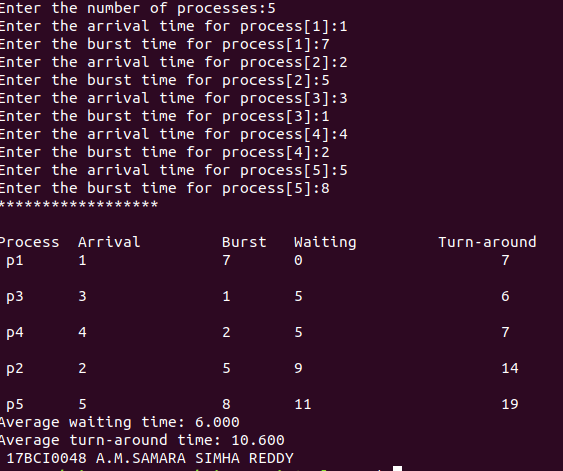
printf("Average turn-around time: %.3f\n",atat);

printf(" 17BCI0048 A.M.SAMARA SIMHA REDDY\n");

return 0;

}

output:



1.

III) SRTF SCHEDULING

code:

#include<stdio.h>

int main()

{

int at[10],bt[10],rt[10],endTime,i,smallest;

int remain=0,n,time,sum\_wait=0,sum\_turnaround=0;

printf("Enter no of Processes : ");

scanf("%d",&n);

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

{

printf("Enter arrival time for Process P%d : ",i+1);

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

printf("Enter burst time for Process P%d : ",i+1);

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

rt[i]=bt[i];

}

printf("\n\nProcess\t|Turnaround Time| Waiting Time\n\n");

rt[9]=9999;

for(time=0;remain!=n;time++)

{

smallest=9;

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

{

if(at[i]<=time && rt[i]<rt[smallest] && rt[i]>0)

{

smallest=i;

}

}

rt[smallest]--;

if(rt[smallest]==0)

{

remain++;

endTime=time+1;

printf("\nP[%d]\t|\t%d\t|\t%d",smallest+1,endTime-at[smallest],endTime-bt[smallest]-at[smallest]);

sum\_wait+=endTime-bt[smallest]-at[smallest];

sum\_turnaround+=endTime-at[smallest];

}

}

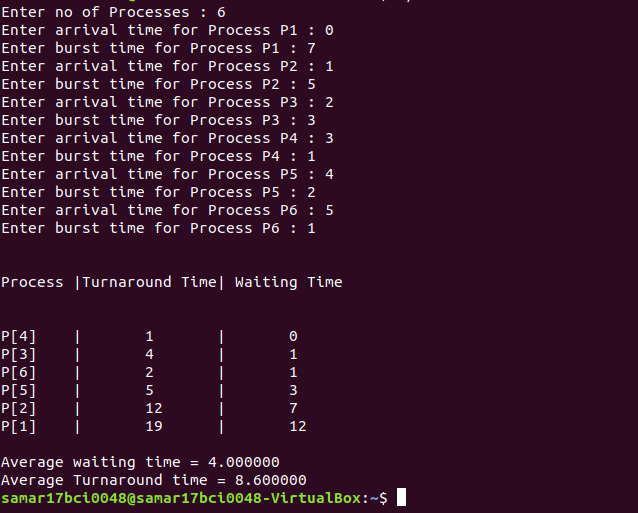
printf("\n\nAverage waiting time = %f\n",sum\_wait\*1.0/n);

printf("Average Turnaround time = %f\n",sum\_turnaround\*1.0/5);

return 0;

}

output:



1.

IV) priority non-preemptive schedulling

code:

#include<stdio.h>

#include<stdlib.h>

typedef struct process{

int at,bt,ct,tat,wt,id,pri;

}process;

process Atthemoment[20];

process hastocome[20];

process completed[20];

process more[20];

process highprior(process more[],int k){

int i,j;

process high=more[0];

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

if(more[i].pri>high.pri){

high=more[i];

}

}

return high;

}

void swap(process \*a,process \*b){

process c=\*a;

\*a=\*b;

\*b=c;

}

int idnotincomp(int id,int d){

int c=0,i,j;

for(i=0;i<d;i++){

if(completed[i].id==id){

c++;

}

}

if(c==0){

return 1;

}

else{

return 0;

}

}

void initatthemoment(){

int i;

for(i=0;i<20;i++){

Atthemoment[i].at='\0';

Atthemoment[i].bt='\0';

Atthemoment[i].ct='\0';

Atthemoment[i].tat='\0';

Atthemoment[i].wt='\0';

}

}

void inithastocome(){

int i;

for(i=0;i<20;i++){

hastocome[i].at='\0';

hastocome[i].bt='\0';

hastocome[i].ct='\0';

hastocome[i].tat='\0';

hastocome[i].wt='\0';

}

}

process shortestarrival(int z){

int i,j,c=0;

process shor=hastocome[0];

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

if(hastocome[i].at<shor.at){

shor=hastocome[i];

}

}

for(i=0;i<z;i++){

if(shor.at==hastocome[i].at){

more[c++]=hastocome[i];

}

}

shor=highprior(more,c);

return shor;

}

process shortest(int m){

int i;

process shor=Atthemoment[0];

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

if(Atthemoment[i].pri>shor.pri){

shor=Atthemoment[i];

}

}

return shor;

}

int main(){

process p[20];

process shor;

int i,j,k,n,m=0,d=0,z,samear=0;

float awt=0,atat=0;

printf("Enter the number of processes:");

scanf("%d",&n);

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

p[i].id=i+1;

printf("Enter the arrival time for process[%d]:",i+1);

scanf("%d",&p[i].at);

printf("Enter the burst time for process[%d]:",i+1);

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

printf("Enter the priority for process[%d]:",i+1);

scanf("%d",&p[i].pri);

}

for(i=0;i<n-1;++i){

for(j=0;j<n-i-1;++j){

if(p[j].at>p[j+1].at){

swap(&p[j],&p[j+1]);

}

}

}

Atthemoment[m++]=p[0];

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

samear=0;

if(p[0].at==p[i].at){

samear++;

Atthemoment[m++]=p[i];

}

}

if(samear>0){

shor=shortest(m);

shor.wt=0;

shor.ct=shor.bt+shor.at;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;}

while(d<n){

m=0;

z=0;

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

if(idnotincomp(p[i].id,d)==1){

if(p[i].at<=completed[d-1].ct){

Atthemoment[m++]=p[i];

}

}

}

if(Atthemoment[0].bt=='\0'){

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

if(idnotincomp(p[i].id,d)==1){

hastocome[z++]=p[i];

}

}

shor=shortestarrival(z);

shor.wt=0;

shor.ct=shor.at+shor.bt;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

inithastocome();

initatthemoment();

}

else{

shor=shortest(m);

shor.wt=completed[d-1].ct-shor.at;

shor.ct=completed[d-1].ct+shor.bt;

shor.tat=shor.ct-shor.at;

completed[d++]=shor;

inithastocome();

initatthemoment();

}

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nProcess\t Arrival\t Priority\t Burst\t Waiting\t Turn-around");

for(i=0;i<d;i++){

printf("\n p%d\t %d\t\t %d\t\t %d\t\t %d\t\t\t%d\n",completed[i].id,completed[i].at,completed[i].pri,completed[i].bt,completed[i].wt,completed[i].tat);

awt=awt+completed[i].wt;

atat=atat+completed[i].tat;

}

awt=awt/n;

atat=atat/n;

printf("Average waiting time: %.3f\n",awt);

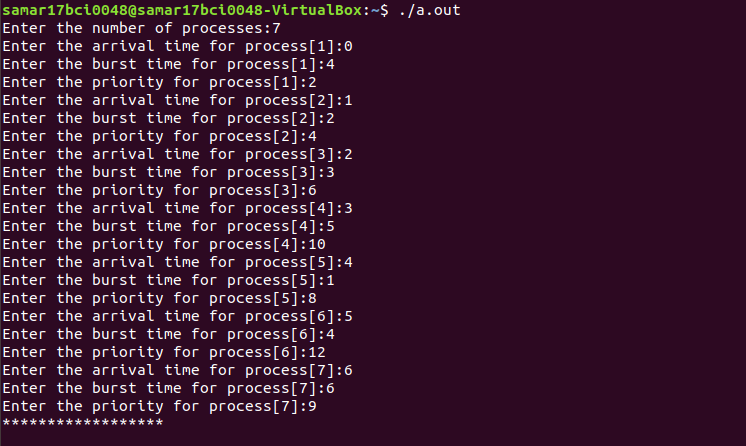
printf("Average turn-around time: %.3f\n",atat);

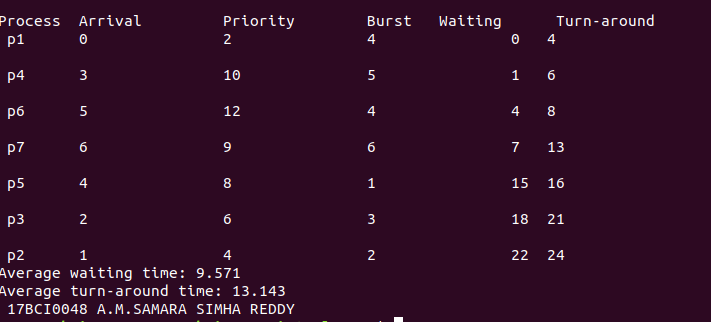
printf(" 17BCI0048 A.M.SAMARA SIMHA REDDY\n");

return 0;

}

output:





1.

v) priority preemptive scheduling

code:

#include<stdio.h>

struct process

{

char process\_name;

int arrival\_time, burst\_time, ct, waiting\_time, turnaround\_time, priority;

int status;

}process\_queue[10];

int limit;

void Arrival\_Time\_Sorting()

{

struct process temp;

int i, j;

for(i = 0; i < limit - 1; i++)

{

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

{

if(process\_queue[i].arrival\_time > process\_queue[j].arrival\_time)

{

temp = process\_queue[i];

process\_queue[i] = process\_queue[j];

process\_queue[j] = temp;

}

}

}

}

int main()

{

int i, time = 0, burst\_time = 0, largest;

char c;

float wait\_time = 0, turnaround\_time = 0, average\_waiting\_time, average\_turnaround\_time;

printf("\nEnter Total Number of Processes:\t");

scanf("%d", &limit);

for(i = 0, c = 'A'; i < limit; i++, c++)

{

process\_queue[i].process\_name = c;

printf("\nEnter Details For Process[%C]:\n", process\_queue[i].process\_name);

printf("Enter Arrival Time:\t");

scanf("%d", &process\_queue[i].arrival\_time );

printf("Enter Burst Time:\t");

scanf("%d", &process\_queue[i].burst\_time);

printf("Enter Priority:\t");

scanf("%d", &process\_queue[i].priority);

process\_queue[i].status = 0;

burst\_time = burst\_time + process\_queue[i].burst\_time;

}

Arrival\_Time\_Sorting();

process\_queue[9].priority = -9999;

printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting Time");

for(time = process\_queue[0].arrival\_time; time < burst\_time;)

{

largest = 9;

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

{

if(process\_queue[i].arrival\_time <= time && process\_queue[i].status != 1 && process\_queue[i].priority > process\_queue[largest].priority)

{

largest = i;

}

}

time = time + process\_queue[largest].burst\_time;

process\_queue[largest].ct = time;

process\_queue[largest].waiting\_time = process\_queue[largest].ct - process\_queue[largest].arrival\_time - process\_queue[largest].burst\_time;

process\_queue[largest].turnaround\_time = process\_queue[largest].ct - process\_queue[largest].arrival\_time;

process\_queue[largest].status = 1;

wait\_time = wait\_time + process\_queue[largest].waiting\_time;

turnaround\_time = turnaround\_time + process\_queue[largest].turnaround\_time;

printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d", process\_queue[largest].process\_name, process\_queue[largest].arrival\_time, process\_queue[largest].burst\_time, process\_queue[largest].priority, process\_queue[largest].waiting\_time);

}

average\_waiting\_time = wait\_time / limit;

average\_turnaround\_time = turnaround\_time / limit;

printf("\n\nAverage waiting time:\t%f\n", average\_waiting\_time);

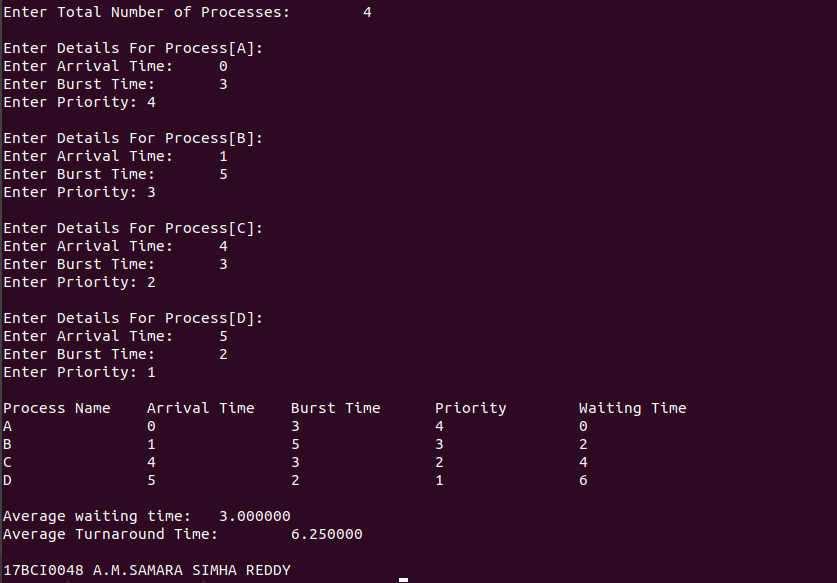
printf("Average Turnaround Time:\t%f\n", average\_turnaround\_time);

printf(" \n17BCI0048 A.M.SAMARA SIMHA REDDY\n");

return 0;

}

output:



vi) round-robin algorithm

code:

#include<stdio.h>

int n;

void SearchStack01(int pnt,int tm);

void SearchStack02(int pnt, int tm);

void AddQue(int pnt);

int at[50], bt[50], ct[50]={0}, qt, rqi[50]={0}, c=0, st, flg=0, tm=0, noe=0, pnt=0, btm[50]={0}, tt, wt;

float att, awt;

int main(){

printf("Enter the number of processes:");

scanf("%d",&n);

for(int x=0;x<n;x++){

printf("\nProcess[%d]",x+1);

printf("\nAT=");

scanf("%d",&at[x]);

printf("BT=");

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

btm[x]=bt[x];}

printf("\nEnter time quantum:");

scanf("%d",&qt);

printf("\nGANTT CHART\n");

printf("%d",at[0]);

do{

if(flg==0){

st=at[0];

//---ReduceBT

if(btm[0]<=qt){

tm=st+btm[0];

btm[0]=0;

SearchStack01(pnt,tm);}

else{

btm[0]=btm[0]-qt;

tm=st+qt;

SearchStack01(pnt,tm);

AddQue(pnt);}

}//if

else{

pnt=rqi[0]-1;

st=tm;

//---DeleteQue

for(int x=0;x<noe && noe!=1;x++){

rqi[x]=rqi[x+1];}

noe--;

//---ReduceBT

if(btm[pnt]<=qt){

tm=st+btm[pnt];

btm[pnt]=0;

SearchStack02(pnt, tm);}

else{

btm[pnt]=btm[pnt]-qt;

tm=st+qt;

SearchStack02(pnt, tm);

AddQue(pnt);}

}//else

//AssignCTvalue

if(btm[pnt]==0){

ct[pnt]=tm;

}//if

flg++;

printf("]-P%d-[%d",pnt+1,tm);

}while(noe!=0);

printf("\n\nPROCESS\t AT\t BT\t CT\t TT\t WT\n");

for(int x=0;x<n;x++){

tt=ct[x]-at[x];

wt=tt-bt[x];

printf("P%d\t%d\t%d\t%d\t%d\t%d\t\n",x+1,at[x],bt[x],ct[x],tt,wt);

awt=awt+wt;

att=att+tt;

}//for

printf("\nAVERAGE TT: %f \nAVERAGE WT: %f",att/n,awt/n);

printf("\n17BCI0048 A.M.SAMARA SIMHA REDDY\n");

return 0;

}//main

void SearchStack01(int pnt,int tm){

for(int x=pnt+1;x<n;x++){

if(at[x]<=tm){

rqi[noe]=x+1;

noe++;}

}//for

}//void

void SearchStack02(int pnt, int tm){

for(int x=pnt+1;x<n;x++){

//---CheckQue

int fl=0;

for(int y=0;y<noe;y++){

if(rqi[y]==x+1){

fl++;}}

if(at[x]<=tm && fl==0 && btm[x]!=0){

rqi[noe]=x+1;

noe++;}

}//for

}//void

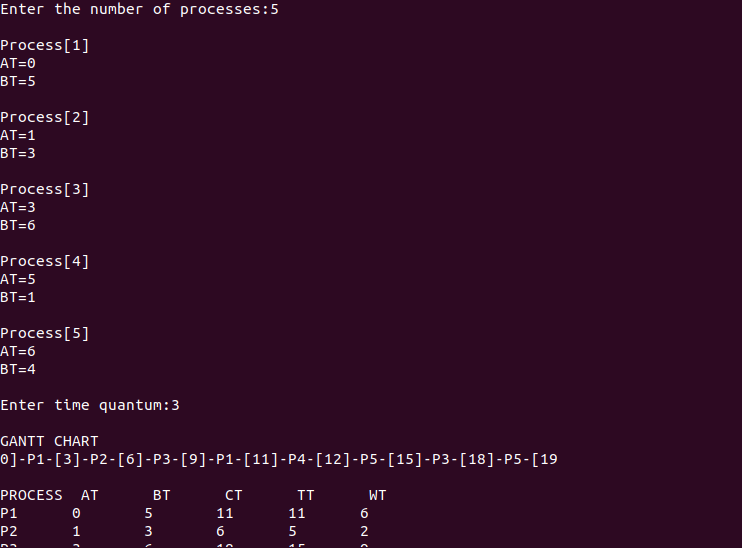
void AddQue(int pnt){

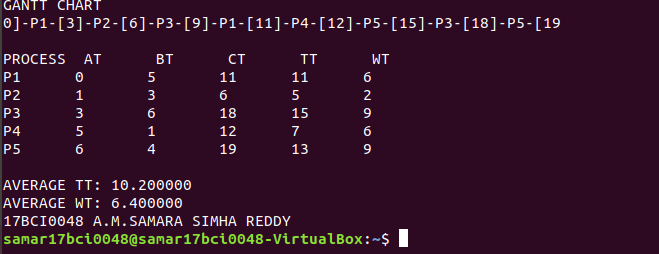
rqi[noe]=pnt+1;

noe++;

}//void

output:





2.

Bankers algorithm

code:

#include<stdio.h>

int main()

{

int k=0,output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10][5],MAX[10][5],pno,P[10],j,rz, count=0;

printf("\n Enter the number of resources : ");

scanf("%d", &rz);

printf("\n enter the max instances of each resources\n");

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

{ avail[i]=0;

printf("%c= ",(i+97));

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

}

printf("\n Enter the number of processes : ");

scanf("%d", &pno);

printf("\n Enter the allocation matrix \n ");

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

printf(" %c",(i+97));

printf("\n");

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

{ P[i]=i;

printf("P[%d] ",P[i]);

for(j=0;j<rz;j++)

{

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

avail[j]+=allocated[i][j];

}

}

printf("\nEnter the MAX matrix \n ");

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

{ printf(" %c",(i+97));

avail[i]=ins[i]-avail[i];

}

printf("\n");

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

{

printf("P[%d] ",i);

for(j=0;j<rz;j++)

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

}

printf("\n");

A: d=-1;

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

{ count=0; t=P[i];

for(j=0;j<rz;j++)

{

need[t][j] = MAX[t][j]-allocated[t][j];

if(need[t][j]<=avail[j])

count++;

}

if(count==rz)

{

output[k++]=P[i];

for(j=0;j<rz;j++)

avail[j]+=allocated[t][j];

}

else

P[++d]=P[i];

}

if(d!=-1)

{ pno=d+1;

goto A;

}

printf("The system is in safe state\n");

printf("\t <");

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

printf(" P[%d] ",output[i]);

printf(">");

printf("\n 17BCI0048 A.M.SAMARA SIMHA REDDY\n");

getchar();

return 0;

}

output:

