1.a) Write a Program to implement FCFS CPU Scheduling algorithm.

**Sourcecode:**

#include<stdio.h>

struct fcfs

{

int at,st,str,ft,tat,wt;

}p[50];

main()

{

int i,j,n;

float atrt=0,awt=0;

clrscr();

printf("\nEnter the number of processes:");

scanf("%d",&n);

printf("\nEnter the arrival times of the processes");

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

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

printf("\nEnter the service times of the processes");

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

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

p[0].str=p[0].at;

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

{

p[j].ft=p[j].str+p[j].st;

p[j+1].str=p[j].ft;

}

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

{

p[i].tat=p[i].ft-p[i].at;

atrt=atrt+p[i].tat;

p[i].wt=p[i].str-p[i].at;

awt=awt+p[i].wt;

}

printf("process\tAT\tST\tSTR\tFT\tTAT\tWT\n");

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

{

printf("p%d\t%d\t%d\t%d\t%d\t%d\t%d\n",i,p[i].at,p[i].st,p[i].str,p[i].ft,p[i].tat,p[i].wt);

}

atrt=atrt/n;

awt=awt/n;

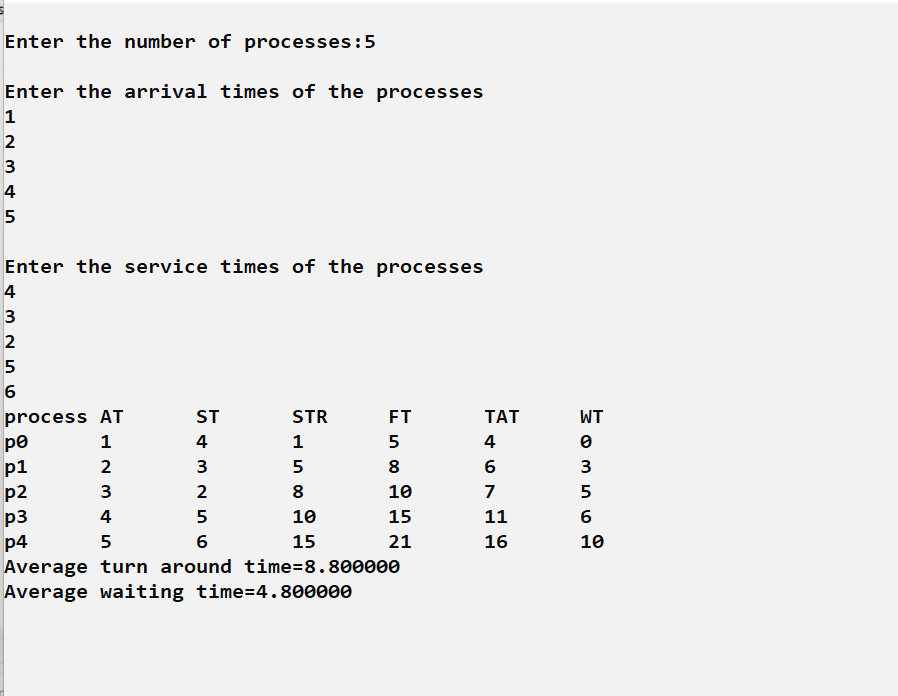
printf("Average turn around time=%f",atrt);

printf("Average waiting time=%f",awt);

getch();

}

**Output:**

****

1.b) Write a Program to implement SJF CPU Scheduling algorithm.

**Sourcecode:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

int et[20],at[10],n,i,j,temp,st[10],ft[10],wt[10],ta[10];

int totwt=0,totta=0;

float awt,ata;

char pn[10][10],t[10];

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

scanf("%d",&n);

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

{

printf("Enter process name, arrival time & service time:");

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

}

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

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

{

if(et[i]<et[j])

{

temp=at[i];

at[i]=at[j];

at[j]=temp;

temp=et[i];

et[i]=et[j];

et[j]=temp;

strcpy(t,pn[i]);

strcpy(pn[i],pn[j]);

strcpy(pn[j],t);

}

}

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

{

if(i==0)

st[i]=at[i];

else

st[i]=ft[i-1];

wt[i]=st[i]-at[i];

ft[i]=st[i]+et[i];

ta[i]=ft[i]-at[i];

totwt+=wt[i];

totta+=ta[i];

}

awt=(float)totwt/n;

ata=(float)totta/n;

printf("\nPname \t arrivaltime \t servicetime \t waitingtime \t tatime");

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

printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],wt[i],ta[i]);

printf("\nAverage waiting time is:%f",awt);

printf("\nAverage turnaroundtime is:%f",ata);

getch();

}

**Output:**

****

**Experiment 2**

**2.a) Write a Program to implement Priority CPU Scheduling algorithm.**

**Sourcecode:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

int et[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];

int totwt=0,totta=0;

float awt,ata;

char pn[10][10],t[10];

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

scanf("%d",&n);

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

{

printf("Enter process name, arrival time, execution time & priority:");

scanf("%s%d%d%d",pn[i],&at[i],&et[i],&p[i]);

}

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

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

{

if(p[i]<p[j])

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

temp=at[i];

at[i]=at[j];

at[j]=temp;

temp=et[i];

et[i]=et[j];

et[j]=temp;

strcpy(t,pn[i]);

strcpy(pn[i],pn[j]);

strcpy(pn[j],t);

}

}

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

{

if(i==0)

{

st[i]=at[i];

wt[i]=st[i]-at[i];

ft[i]=st[i]+et[i];

ta[i]=ft[i]-at[i];

}

else

{

st[i]=ft[i-1];

wt[i]=st[i]-at[i];

ft[i]=st[i]+et[i];

ta[i]=ft[i]-at[i];

}

totwt+=wt[i];

totta+=ta[i];

}

awt=(float)totwt/n;

ata=(float)totta/n;

printf("\nPname \t arrivaltime \t executiontime \t priority \t waitingtime \t tatime");

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

printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],p[i],wt[i],ta[i]);

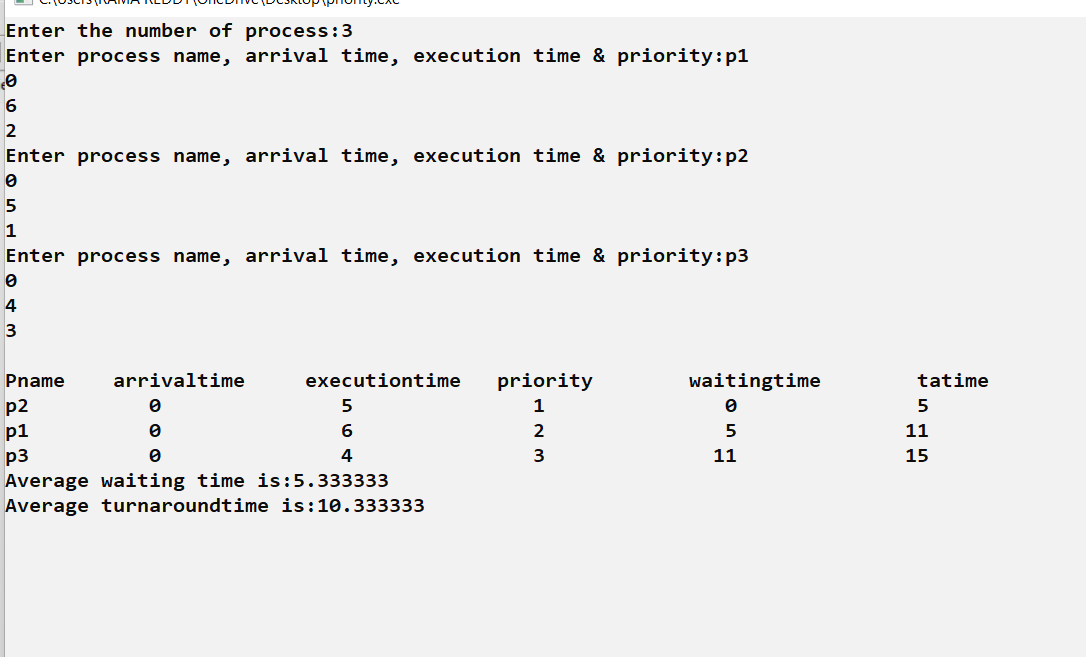
printf("\nAverage waiting time is:%f",awt);

printf("\nAverage turnaroundtime is:%f",ata);

getch();

}

**Output:**

****

**2.b) Write a Program to implement Round Robin CPU Scheduling algorithm.**

**Sourcecode:**

#include<stdio.h>

#include<conio.h>

main()

{

int b[10], pno[10],ts,n,s[10],e[10],w[10],t[10],r[10];

int i,c=0,x=0;

float aw=0,at=0;

printf(“Enter number of processes”);

scanf(“%d”,&n);

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

pno[i]=i+1;

printf(“Enter the time slice”);

scanf(“%d”,&ts);

printf(“Enter the burst time of each process”);

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

scanf(“%d”,&b[i]);

s[0]=0;

x=0;

c=0;

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

{

if(b[i]<ts)

{

e[i]=x+b[i];

r[i]=0;

}

else

{

e[i]=ts+x;

r[i]=b[i]-ts;

}

x=e[i];

s[i+1]=e[i];

t[i]=e[i];

w[i]=s[i];

}

while(c>=0)

{

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

{

if(r[i]!=0)

{

w[i]=w[i]+x-e[i];

if(r[i]<ts)

{

e[i]=x+r[i];

r[i]=0;

}

else

{

e[i]=x+ts;

r[i]=r[i]-ts;

}

x=e[i];

t[i]=e[i];

}

if(r[i]!=0)

c++;

}

c--;

}

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

{

aw=aw+w[i];

at=at+t[i];

}

aw=aw/n;

at=at/n;

printf(“Time slice=%d”,ts);

printf(“\npno \tbt \tst \tet \twt \ttat”);

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

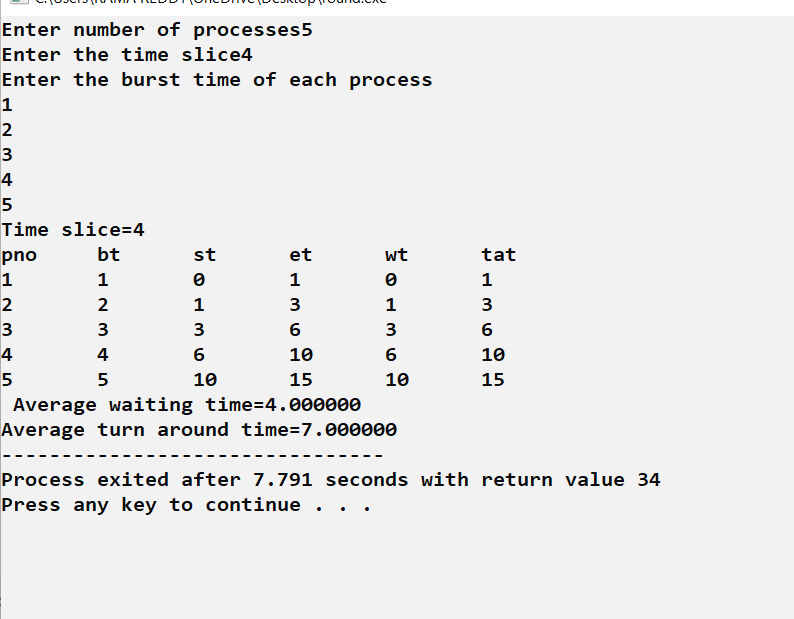
printf(“\n%d\t%d\t%d\t%d\t%d\t%d”,pno[i],b[i],s[i],e[i],w[i],t[i]);

printf(“\n Average waiting time=%f”,aw);

printf(“\nAverage turn around time=%f”,at);

}

**Output:**

****

**Experiment-3**

**3.a) Implementation of fork (), wait (), exec (), and exit (), System calls**.

**Sourcecode:**

#include<stdio.h>

#include<sys/types.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

pid\_t p;

int i;

p=fork();

if(p==1)

{

printf("fork error");

exit(-1);

}

else if(p==0)

{

for(i=0;i<5;i++)

{

execlp("ls","ls","-l",NULL);

printf("child process\n");

}

}

else

{

wait(0);

for(i=0;i<5;i++)

{

printf("parent process\n");

}

exit(0);

}

}

**Output:**

**Experiment 4**

**4) Write a Program to implement Multiprogramming with a fixed number of tasks (MFT)**

**Sourcecode:**

#include<stdio.h>

#include<math.h>

main()

{

int np,nb,mm,bs,i,j,ps[100],nba[100],ifm[100],sb=0,flag=0;

float x;

printf("Enter the Memory size");

scanf("%d",&mm);

printf("Enter the no of Blocks");

scanf("%d",&nb);

printf("Enter the no of processes");

scanf("%d",&np);

bs=mm/nb;

for(i=1;(i<=np)&&(sb<nb);i++)

{

printf("Enter the size of p[%d]:",i);

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

if(ps[i]<=bs)

nba[i]=1;

else

{

x=ps[i]/(float)bs;

nba[i]=(ceil)(x);

}

ifm[i]=nba[i]\*bs-ps[i];

sb=sb+nba[i];

if(sb>nb)

{

i=i-1;

flag=1;

}

}

j=i;

printf("Process\tSize\tnba\tifm \n");

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

printf("%d\t%d\t%d\t%d\n", i, ps[i], nba[i], ifm[i]);

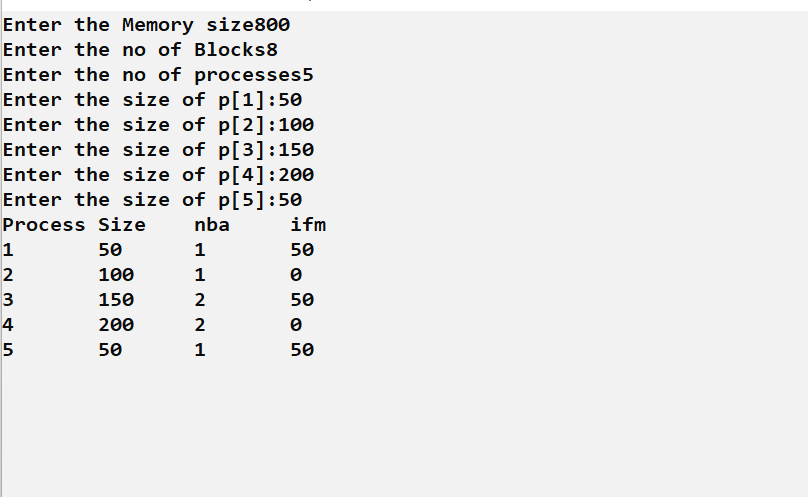
if(flag==1)

printf("Memory space is unavailable");

getch();

}

**Output:**



**Experiment 5**

**4) Write a Program to implement Multiprogramming with a variable number of tasks (MVT)**

**Sorcecode:**

#include<stdio.h>

main()

{

int mm,np,ps[100],rm[100],am=0,flag=0,i,j;

printf(“Enter the memory size”);

scanf(“%d”,&mm);

printf(“enter no of processes”);

scanf(“%d”,&np);

for(i=0;(i<np)&&(am<mm);i++)

{

printf(“Enter the size of p[%d]:”,i+1);

scanf(“%d”,&ps[i]);

am=am+ps[i];

if(am>=mm)

{

flag=1;

break;

}

rm[i]=mm-am;

}

j=i;

printf(“Process \t size \t rm \n”);

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

printf(“%d \t %d \t %d \n ”, i+1, ps[i] , rm[i]);

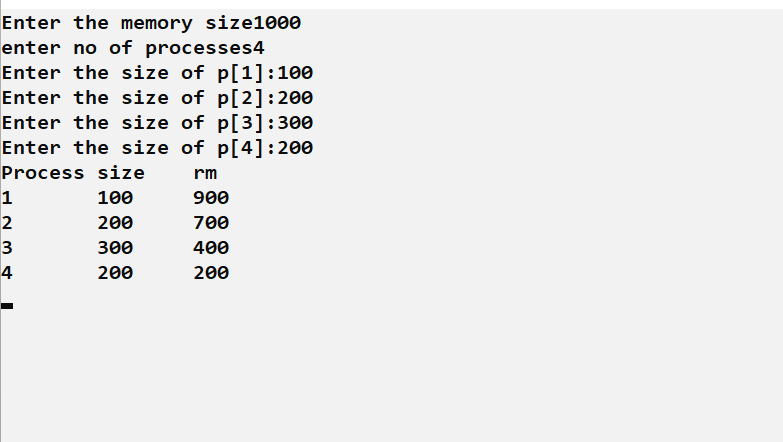
if(flag==1)

printf(“memory is unavailable”);

getch();

}

**Output:**

****