RAM KRISHNA COLLEGE

MADHUBANI-847211(BIHAR)

(A Constituent Unit of L.N.MITHILA UNIVERSITY, DARBHANGA)

Department of B.C.A.(H)

Year: 2rd **SESSION:2021-2024** **Semester:3rd**

Operating System Lab

LAB MANUAL

Prepared By:

CO-ORDINATOR (B.C.A)

RAM KRISHNA COLLEGE

MADHUBANI-847211(BIHAR)

(A Constituent Unit of L.N.MITHILA UNIVERSITY, DARBHANGA)

Department of B.C.A.(H)

INDEX

S.No	Practical's Name	S/w used	Remark
1	**A program to simulate the FCFS CPU scheduling algorithm**	Turbo C/C++	
2	**A program to simulate the SJF CPU scheduling algorithm**	Turbo C/C++	
3	**A program to simulate the priority CPU scheduling algorithm**	Turbo C/C++	
4	**A program to simulate the Round Robin CPU scheduling algorithm**	Turbo C/C++	
5	**A program to simulate the MVT.**	Turbo C/C++	
6	**A Program to simulate the MFT**	Turbo C/C++	
7	**A program to simulate the Bankers Algorithm for Deadlock Avoidance.**	Turbo C/C++	
8	**A program to simulate Bankers Algorithm for Deadlock Prevention.**	Turbo C/C++	
9	**A program to simulate FIFO Page Replacement Algorithm**	Turbo C/C++	
10	**A program to simulate LRU Page Replacement Algorithm**	Turbo C/C++	

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 1**	
	Course Code : **Faculty :**	**Branch: B.C.A(H)**	**Semester: 3rd**

OBJECTIVE: A program to simulate the FCFS CPU scheduling algorithm

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char pn[10][10];
int arr[10],bur[10],star[10],finish[10],tat[10],wt[10],i,n;
int totwt=0,tottat=0;
clrscr();
printf("Enter the number of processes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the Process Name, Arrival Time & Burst Time:");
scanf("%s%d%d",&pn[i],&arr[i],&bur[i]);
}
for(i=0;i<n;i++)
{
if(i==0)
{
star[i]=arr[i];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
else
{
star[i]=finish[i-1];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
}
printf("\nPName    Arrtime    Burtime    Start    TAT  Finish");
```

```
for(i=0;i<n;i++)
{
printf("\n%s\t%6d\t\t%6d\t%6d\t%6d\t%6d",pn[i],arr[i],bur[i],star[i],tat[i],finish[i]);
totwt+=wt[i];
tottat+=tat[i];
}
printf("\nAverage Waiting time:%f",(float)totwt/n);
printf("\nAverage Turn Around Time:%f",(float)tottat/n);
getch();
}
```

OUTPUT:

Input:

Enter the number of processes: 3
Enter the Process Name, Arrival Time & Burst Time: 1 2 3
Enter the Process Name, Arrival Time & Burst Time: 2 5 6
Enter the Process Name, Arrival Time & Burst Time: 3 6 7

Output:

PName	Arrtime	Burtime	Srart	TAT	Finish
1	2	3	2	3	5
2	5	6	5	6	4
3	6	7	6	7	10

Average Waiting Time: 3.333
Average Turn Around Time: 7.000

Outcome:
1) To understand the simulation process of FCFS CPU scheduling algorithm.

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 2**	
	Course Code : **Faculty :**	**Branch: B.C.A.(H)**	**Semester: 3rd**

OBJECTIVE: A program to simulate the SJF CPU scheduling algorithm

PROGRAM:

```
#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];
clrscr();
printf("Enter the number of process:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter process name, arrival time & execution time:");
flushall();
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\tarrivaltime\texecutiontime\twaitingtime\ttatime");
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:

Input:

Enter the number of processes: 3
Enter the Process Name, Arrival Time & Burst Time: 1 4 6
Enter the Process Name, Arrival Time & Burst Time: 2 5 15
Enter the Process Name, Arrival Time & Burst Time: 3 6 11

Output:

Pname	arrivaltime	executiontime	waitingtime	tatime
1	4	6	0	6
3	6	11	4	15
2	5	15	16	31

Average Waiting Time: 6.6667
Average Turn Around Time: 17.3333

Outcome:

1) To understand the simulation process of SJF CPU scheduling algorithm.

OBJECTIVE: A program to simulate the priority CPU scheduling algorithm

PROGRAM:

```
#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];
clrscr();
printf("Enter the number of process:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter process name,arrivaltime,execution time & priority:");
flushall();
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\tarrivaltime\texecutiontime\tpriority\twaitingtime\ttatime");
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:

Input:

```
Enter the number of processes: 3
Enter the Process Name, Arrival Time, execution time & priority: 1 2 3 1
Enter the Process Name, Arrival Time, execution time & priority: 2 4 5 2
Enter the Process Name, Arrival Time, execution time & priority: 3 5 6 3
```

Output:

```
Pname    arrivaltime   executiontime   priority   waitingtime      tatime
1                2                3          1             0                3
```

2	4	5	2	1	6
3	5	6	3	5	11

Average Waiting Time: 2.0000
Average Turn Around Time: 6.6667

Outcome:

1) To understand the process to simulate the priority CPU scheduling algorithm.

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 4**	
	Course Code : **Faculty :**	**Branch:B.C.A(H)**	**Semester: 3rd**

OBJECTIVE: A program to simulate the Round Robin CPU scheduling algorithm

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int et[30],ts,n,i,x=0,tot=0;
char pn[10][10];
clrscr();
printf("Enter the no of processes:");
scanf("%d",&n);
printf("Enter the time quantum:");
scanf("%d",&ts);
for(i=0;i<n;i++)
{
printf("enter process name  & estimated time:");
scanf("%s %d",pn[i],&et[i]);
}
printf("The processes are:");
for(i=0;i<n;i++)
printf("process %d: %s\n",i+1,pn[i]);
for(i=0;i<n;i++)
tot=tot+et[i];
while(x!=tot)
{
for(i=0;i<n;i++)
{
if(et[i]>ts)
{
x=x+ts;
printf("\n %s -> %d",pn[i],ts);

et[i]=et[i]-ts;
```

```
}

else
if((et[i]<=ts)&&et[i]!=0)
{
x=x+et[i];
printf("\n %s  ->  %d",pn[i],et[i]);
et[i]=0;}
}
}
printf("\n Total Estimated Time:%d",x);
getch();
}
```

OUTPUT:

Input:

Enter the no of processes: 2
Enter the time quantum: 3

Enter the process name & estimated time: p1 12
Enter the process name & estimated time: p2 15

Output:

p1 -> 3
p2 -> 3
p1 -> 3
p2 -> 3
p1 -> 3
p2 -> 3
p1 -> 3
p2 -> 3
p2 -> 3

Total Estimated Time: 27

Outcome:

1) To understand the simulation process of the Round Robin CPU scheduling algorithm

OBJECTIVE: A program to simulate the MVT.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m=0,m1=0,m2=0,p,count=0,i;
clrscr();
printf("enter the memory capacity:");
scanf("%d",&m);
printf("enter the no of processes:");
scanf("%d",&p);
for(i=0;i<p;i++)
{
printf("\nenter memory req for process%d: ",i+1);
scanf("%d",&m1);
count=count+m1;
if(m1<=m)
{
if(count==m)
{
printf("there is no further memory remaining:");
}
else
{
printf("the memory allocated for process%d is: %d ",i+1,m);
m2=m-m1;
printf("\nremaining memory is: %d",m2);
m=m2;
}
}
```

```
else
{
printf("memory is not allocated for process%d",i+1);
}
printf("\nexternal fragmentation for this process is:%d",m2);
}
getch();
```

Outcome:

1) To understand the MVT simulation process.

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 6**	
	Course Code : **Faculty :**	**Branch: B.C.A(H)**	**Semester: 3rd**

OBJECTIVE: A Program to simulate the MFT

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m,p,s,p1;
int m1[4],i,f,f1=0,f2=0,fra1,fra2;
clrscr();
printf("Enter the memory size:");
scanf("%d",&m);
printf("Enter the no of partitions:");
scanf("%d",&p);
s=m/p;
printf("Each partn size is:%d",s);
printf("\nEnter the no of processes:");
scanf("%d",&p1);
for(i=0;i<p1;i++)
{
printf("\nEnter the memory req for process%d:",i+1);
scanf("%d",&m1[i]);
if(m1[i]<=s)
{
printf("\nProcess is allocated in partition%d",i+1);
fra1=s-m1[i];
printf("\nInternal fragmentation for process is:%d",fra1);
f1=f1+fra1;
}
else
{
printf("\nProcess not allocated in partition%d",i+1);
fra2=s;
f2=f2+fra2;
printf("\nExternal fragmentation for partition is:%d",fra2);
```

```
}
}
printf("\nProcess\tmemory\tallocatedmemory");
for(i=0;i<p1;i++)
printf("\n%5d\t%5d\t%5d",i+1,s,m1[i]);
f=f1+f2;
printf("\nThe tot no of fragmentation is:%d",f);
getch();
}
```

Outcome:

1) To understand the simulation process of MFT

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 7**	
	Course Code : **Faculty :**	**Branch: B.C.A(H)**	**Semester: 3rd**

OBJECTIVE: A program to simulate the Bankers Algorithm for Deadlock Avoidance.

PROGRAM:

```
//Bankers algorithm for deadlock avoidance.
#include<stdio.h>
#include<conio.h>
void main()
{
int n,r,i,j,k,p,u=0,s=0,m;
int block[10],run[10],active[10],newreq[10];
int max[10][10],resalloc[10][10],resreq[10][10];
int totalloc[10],totext[10],simalloc[10];
clrscr();
printf("Enter the no of processes:");
scanf("%d",&n);
printf("Enter the no of resource classes:");
scanf("%d",&r);
printf("Enter the total existed resource in each class:");
for(k=1;k<=r;k++)
scanf("%d",&totext[k]);
printf("Enter the allocated resources:");
for(i=1;i<=n;i++)
for(k=1;k<=r;k++)
scanf("%d",&resalloc);
printf("Enter the process making the new request:");
scanf("%d",&p);
printf("Enter the requested resource:");
for(k=1;k<=r;k++)
scanf("%d",&newreq[k]);
printf("Enter the process which are n blocked or running:");
for(i=1;i<=n;i++)
{
```

```
if(i!=p)
{
printf("process %d:\n",i+1);
scanf("%d%d",&block[i],&run[i]);
}
}
block[p]=0;
run[p]=0;
for(k=1;k<=r;k++)
{
j=0;
for(i=1;i<=n;i++)
{
totalloc[k]=j+resalloc[i][k];
j=totalloc[k];
}
}
for(i=1;i<=n;i++)
{
if(block[i]==1||run[i]==1)
active[i]=1;
else
active[i]=0;
}
for(k=1;k<=r;k++)
{
resalloc[p][k]+=newreq[k];
totalloc[k]+=newreq[k];
}
for(k=1;k<=r;k++)
{
if(totext[k]-totalloc[k]<0)
{
u=1;break;
}
}
if(u==0)
{
for(k=1;k<=r;k++)
simalloc[k]=totalloc[k];
for(s=1;s<=n;s++)
for(i=1;i<=n;i++)
{
if(active[i]==1)
{
j=0;
```

```
for(k=1;k<=r;k++)
{
if((totext[k]-simalloc[k])<(max[i][k]-resalloc[i][k]))
{
j=1;break;
}
}
}
if(j==0)
{
active[i]=0;
for(k=1;k<=r;k++)
simalloc[k]=resalloc[i][k];
}
}
m=0;
for(k=1;k<=r;k++)
resreq[p][k]=newreq[k];
printf("Deadlock willn't occur");
}
else
{
for(k=1;k<=r;k++)
{
resalloc[p][k]=newreq[k];
totalloc[k]=newreq[k];
}
printf("Deadlock will occur");
}
getch();
}
```

Outcome:

1) To understand the simulation process of Bankers Algorithm for Deadlock Avoidance.

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 8**	
	Course Code : **Faculty :**	**Branch: B.C.A(H)**	**Semester: 3rd**

OBJECTIVE: A program to simulate Bankers Algorithm for Deadlock Prevention.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int cl[10][10],al[10][10],av[10],i,j,k,m,n,c,ne[10][10],flag=0;
clrscr();
printf("\nEnter the matrix");
scanf("%d %d",&m,&n);
printf("\nEnter the clOBJECTIVE matrix");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&cl[i][j]);
}
}
printf("\nEnter allocated matrix");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&al[i][j]);
}
}
printf("\nThe need matrix");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
ne[i][j]=cl[i][j]-al[i][j];
printf("\t%d",ne[i][j]);
```

```
}
printf("\n");
}
printf("\nEnter avaliable matrix");
for(i=0;i<3;i++)
scanf("%d",av[i]);
printf("ClOBJECTIVE matrix:\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("\t%d",cl[i][j]);
}
printf("\n");
}
printf("\n allocated matrix:\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("\t%d",al[i][j]);
}
printf("\n");
}
printf(" available matrix:\n");
for(i=0;i<3;i++)
{
printf("\t%d",av[i]);
}
for(k=0;k<m;k++)
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
if(av[j]>=ne[i][j])
flag=1;
else
break;
if(flag==1&& j==n-1)
goto a;
}
}
a: if(flag==0)
{
printf("unsafestate");
}
```

```
if(flag==1)
{
flag=0;
for(i=0;i<m;i++)
{
for(j=0;j<n;i++)
{
av[j]+=al[i][j];
al[i][j]=1;
}
}
printf("\n safe state");
for(i=0;i<n;i++)
printf("\t available matrix:%d",av[i]);
}
getch();
}
```

Outcome:

1) To understand the simulation process of Bankers Algorithm for Deadlock Prevention.

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 9**	
	Course Code : **Faculty :**	**Branch: B.C.A(H)**	**Semester: 3rd**

OBJECTIVE: A program to simulate FIFO Page Replacement Algorithm

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],b[20],p=0,q=0,m=0,h,k,i,q1=1,j,u;
char f='F';
clrscr();
printf("Enter numbers:");
for(i=0;i<12;i++)
scanf("%d",&b[i]);
for(i=0;i<12;i++)
{if(p==0)
{
if(q>=3)
q=0;
a[q]=b[i];
q++;
if(q1<3)
{
q1=q;
}
}
printf("\n%d",b[i]);
printf("\t");
for(h=0;h<q1;h++)
printf("%d",a[h]);
if((p==0)&&(q1==3))
```

```
{
printf("-->%c",f);
m++;
}
p=0;
for(k=0;k<q-1;k++)
{
if(b[i+1]==a[k])
p=1;
}
}
printf("\nNo of faults:%d",m);
getch();
}
```

Output

```
ENTER THE NUMBER OF PAGES:  20
ENTER THE PAGE NUMBER :      7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
ENTER THE NUMBER OF FRAMES :3
   ref string      page frames
7          7     -1    -1
0          7     0     -1
1          7     0     1
2          2     0     1
0
3          2     3     1
0          2     3     0
4          4     3     0
2          4     2     0
3          4     2     3
0          0     2     3
3
2
1          0     1     3
2          0     1     2
0
1
7          7     1     2
0          7     0     2
1          7     0     1
Page Fault Is 15
```

Outcome:

1) A program to simulate FIFO Page Replacement Algorithm

RAM KRISHNA COLLEGE

Department of B.C.A.(H)

LAB MANUAL

	Course Name : Operating System Lab.	**EXPERIMENT NO. 10**	
	Course Code : **Faculty :**	**Branch: B.C.A(H)**	**Semester: 3 rd**

OBJECTIVE: A program to simulate LRU Page Replacement Algorithm

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int g=0,a[5],b[20],p=0,q=0,m=0,h,k,i,q1=1,j,u;
char f='F';
clrscr();
printf("Enter no:");
for(i=0;i<12;i++)
scanf("%d",&b[i]);
for(i=0;i<12;i++)
{if(p==0)
{
if(q>=3)
q=0;
a[q]=b[i];
q++;
if(q1<3)
{
q1=q;
g=1;
}
}
printf("\n%d",b[i]);
printf("\t");
for(h=0;h<q1;h++)
printf("%d",a[h]);
if((p==0)&&(q1==3)&&(g!=1))
```

```
{
printf("-->%c",f);
m++;
}
p=0;
g=0;
if(q1==3)
{
for(k=0;k<q-1;k++)
{
if(b[i+1]==a[k])
p=1;
}
for(j=0;j<q1;j++)
{
u=0;
k=i;
while(k>(i-2)&&(k>=0))
{
if(b[k]==a[j])
u++;
k--;
}
if(u==0)
q=j;
}
}
else
{
for(k=0;k<q;k++)
{
if(b[i+1]==a[k])
p=1;
}
}
}
printf("\nNo of faults:%d",m);
getch();

}
```

OUTPUT

```
Enter no of pages:10
Enter the reference string:7 5 9 4 3 7 9 6 2 1
Enter no of frames:3
      7
      7      5
```

7	5	9
4	5	9
4	3	9
4	3	7
9	3	7
9	6	7
9	6	2
1	6	2

The no of page faults is 10

Outcome:

1) To understand the Simulation process of LRU Page Replacement Algorithm.