Operating Systems Lab

1. Simulate the Following CPU Scheduling Algorithms

A) FCFS Scheduling :

```
#include<stdio.h>
#include<conio.h>
main()
{
int bt[20], wt[20], tat[20], i, n;
 float wtavg, tatavg;
clrscr();
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\nEnter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
        printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
getch();
}
```

Output :

Enter the number of processes -- 3
Enter Burst Time for Process 0 -- 24
Enter Burst Time for Process 1 -- 3
Enter Burst Time for Process 2 -- 3

PROCESS	BURST TIME	WAITING TIME	TURNAROUNDTIME
P0	24	0	24
P1	3	24	27
P2	3	27	30

Average Waiting Time-- 17.000000

Average Turnaround Time -- 27.000000

B) SJF Scheduling :

```
#include<stdio.h>
#include<conio.h>
main()
{
int p[20], bt[20], wt[20], tat[20], i, k, n, temp; float wtavg,
tatavg;
clrscr();
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
p[i]=i;
printf("Enter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);

}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(bt[i]>bt[k])
{
temp=bt[i];
bt[i]=bt[k];
bt[k]=temp;

temp=p[i];
p[i]=p[k];
p[k]=temp;
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0]; for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
 tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
        printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n); getch();
```

}

Output :

Enter the number of processes -- 4
Enter Burst Time for Process 0 -- 6
Enter Burst Time for Process 1 -- 8
Enter Burst Time for Process 2 -- 7
Enter Burst Time for Process 3 -- 3

PROCESS	BURST TIME	WAITING TIME	TURNAROUND TIME
P3	3	0	3
P0	6	3	9
P2	7	9	16
P1	8	16	24

Average Waiting Time -- 7.000000
Average Turnaround Time -- 13.000000

C) Priority Scheduling :

```
#include<stdio.h>
main()
{
int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp; float wtavg,
tatavg;
clrscr();
printf("Enter the number of  processes --- ");
scanf("%d",&n);
for(i=0;i<n;i++){
p[i] = i;
printf("Enter the Burst Time & Priority of Process %d --- ",i); scanf("%d
%d",&bt[i], &pri[i]);
}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(pri[i] > pri[k]){
temp=p[i];
p[i]=p[k];
p[k]=temp;
temp=bt[i];
bt[i]=bt[k];
bt[k]=temp;
temp=pri[i];
pri[i]=pri[k];
pri[k]=temp;
}
wtavg = wt[0] = 0;
tatavg = tat[0] = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] + bt[i-1];
tat[i] = tat[i-1] + bt[i];

wtavg = wtavg + wt[i];
 tatavg = tatavg + tat[i];
}
printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND
TIME");
for(i=0;i<n;i++)
printf("\n%d \t\t %d \t\t %d \t\t %d \t\t %d ",p[i],pri[i],bt[i],wt[i],tat[i]);
printf("\nAverage Waiting Time is --- %f",wtavg/n); printf("\nAverage
Turnaround Time is --- %f",tatavg/n);
getch();
}
```

Output :

```
Enter the number of processes -- 5
Enter the Burst Time & Priority of Process 0 --- 10        3
Enter the Burst Time & Priority of Process 1 --- 1         1
Enter the Burst Time & Priority of Process 2 --- 2         4
Enter the Burst Time & Priority of Process 3 --- 1         5
Enter the Burst Time & Priority of Process 4 --- 5         2

PROCESS     PRIORITY     BURST TIME     WAITINGTIME     TURNAROUND TIME

1           1            1
                                        0               1
4           2            5              1               6
0           3            10             6               16
2           4            2              16              18
3           5            1              18              19
Average Waiting Time is ---          8.200000
Average Turnaround Time is ---------- 12.000000
```

D) Round Robin Scheduling :

```
#include<stdio.h>
main()
{
int    i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
float awt=0,att=0,temp=0;
clrscr();
printf("Enter the no of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter Burst Time for process %d -- ", i+1);
scanf("%d",&bu[i]);
ct[i]=bu[i];
}
printf("\nEnter the size of time slice -- ");
scanf("%d",&t);
max=bu[0];
for(i=1;i<n;i++)
if(max<bu[i])
max=bu[i];
for(j=0;j<(max/t)+1;j++)
for(i=0;i<n;i++)
if(bu[i]!=0)
 if(bu[i]<=t)    {
```

```
tat[i]=temp+bu[i];
temp=temp+bu[i];
bu[i]=0;
 }
 else {
bu[i]=bu[i]-t;
temp=temp+t;
 }
for(i=0;i<n;i++){
wa[i]=tat[i]-
ct[i]; att+=tat[i];
awt+=wa[i];}
printf("\nThe Average Turnaround time is -- %f",att/n);
printf("\nThe Average Waiting time is -- %f ",awt/n);
printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);getch();
}
```

Output :

Enter the no of processes – 3
Enter Burst Time for process 1 – 24
Enter Burst Time for process 2 -- 3
Enter Burst Time for process 3 – 3
Enter the size of time slice – 3

PROCESS	BURST TIME	WAITING TIME	TURNAROUNDTIME
1	24	6	30
2	3	4	7
3	3	7	10

The Average Turnaround time is – 15.666667 The
Average Waiting time is ---------- 5.666667

2. Simulate The Following
a. Multiprogramming with A Fixed Number Of Tasks (MFT)

```
#include<stdio.h>
#include<conio.h>
main()
{
int ms, bs, nob, ef,n,
mp[10],tif=0; int i,p=0;
clrscr();
printf("Enter the total memory available (in Bytes) -- ");
scanf("%d",&ms);
printf("Enter the block size (in Bytes) -- ");
scanf("%d", &bs);
nob=ms/bs;
ef=ms - nob*bs;
printf("\nEnter the number of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter memory required for process %d (in Bytes)-- ",i+1);
scanf("%d",&mp[i]);
}
printf("\nNo. of Blocks available in memory--%d",nob);
printf("\n\nPROCESS\tMEMORYREQUIRED\tALLOCATED\tINTERNAL
FRAGMENTATION");
for(i=0;i<n && p<nob;i++)
{
printf("\n %d\t\t%d",i+1,mp[i]);
if(mp[i] > bs)
printf("\t\tNO\t\t---");
else
{
printf("\t\tYES\t%d",bs-mp[i]);
 tif = tif + bs-mp[i];
p++;
}
}
if(i<n)
printf("\nMemory is Full, Remaining Processes cannot be accomodated");
printf("\n\nTotal Internal Fragmentation is %d",tif);
 printf("\nTotal External Fragmentation is %d",ef);
getch();
}
```

Output :

Enter the total memory available (in Bytes) -- 1000

Enter the block size (in Bytes)-- 300

Enter the number of processes – 5

Enter memory required for process 1 (in Bytes) -- 275

Enter memory required for process 2 (in Bytes) -- 400

Enter memory required for process 3 (in Bytes) -- 290

Enter memory required for process 4 (in Bytes) -- 293

Enter memory required for process 5 (in Bytes) -- 100

No. of Blocks available in memory -- 3

PROCESS	MEMORY REQUIRED	ALLOCATED	INTERNAL FRAGMENTATION
1	275	YES	25
2	400	NO	-----
3	290	YES	10
4	293	YES	7

Memory is Full, Remaining Processes cannot be accommodated Total Internal Fragmentation is 42

Total External Fragmentation is 100

b. Multiprogramming with A Variable Number Of Tasks (MVT)

```
#include<stdio.h>
#include<conio.h>
main()
 {
 int              ms,mp[10],i,
 temp,n=0; char ch = 'y';
 clrscr();
 printf("\nEnter the total memory available (in Bytes)-- ");
 scanf("%d",&ms);
 temp=ms;
 for(i=0;ch=='y';i++,n++)
 {
 printf("\nEnter memory required for process %d (in Bytes) -- ",i+1);
 scanf("%d",&mp[i]);
 if(mp[i]<=temp)
 {
 printf("\nMemory is allocated for Process %d ",i+1);
 temp = temp - mp[i];
 }
 else
 {
 printf("\nMemory is Full"); break;
 }
 printf("\nDo you want to continue(y/n) -- ");
  scanf(" %c", &ch);
 }
 printf("\n\nTotal Memory Available -- %d", ms);
 printf("\n\n\tPROCESS\t\t MEMORY ALLOCATED ");
 for(i=0;i<n;i++)
 printf("\n \t%d\t\t%d",i+1,mp[i]);
 printf("\n\nTotal Memory Allocated is %d",ms-temp);
 printf("\nTotal External Fragmentation is %d",temp);
 getch();
 }
```

Output :

Enter the total memory available (in Bytes) – 1000
Enter memory required for process 1 (in Bytes) – 400
Memory is allocated for Process 1
Do you want to continue(y/n) -- y
Enter memory required for process 2 (in Bytes) -- 275
Memory is allocated for Process 2
Do you want to continue(y/n) – y
Enter memory required for process 3 (in Bytes) – 550

Memory is Full

Total Memory Available – 1000

PROCESS	MEMORY ALLOCATED
1	400
2	275

Total Memory Allocated is 675
Total External Fragmentation is 325

3. Write a program to implement first fit, best fit and worst fit algorithm for memory management.

FIRST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
        int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
        static int bf[max],ff[max];
        clrscr();
        printf("\n\tMemory Management Scheme - Worst Fit");
        printf("\nEnter the number of blocks:");
         scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
         for(i=1;i<=nb;i++)
        {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
        }
        printf("Enter the size of the files :-\n");
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
```

```
            for(i=1;i<=nf;i++)
            {
                    for(j=1;j<=nb;j++)
                    {
                            if(bf[j]!=1)          //if bf[j] is not allocated
                            {
                                    temp=b[j]-f[i];
                                    if(temp>=0)
                                            if(highest<temp)
                                            {
                            }
                    }
                     frag[i]=highest; bf[ff[i]]=1; highest=0;
            }
            ff[i]=j; highest=temp;
          }
            printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement");
            for(i=1;i<=nf;i++)
                    printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
            getch();
}
```

Output :

Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:-
Block 1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:-
File 1: 1
File 2: 4

File No	File Size	Block No	Block Size	Fragment
1	1	3	7	6
2	4	1	5	1

BEST-FIT

```
 #include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
          int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
          static int bf[max],ff[max];
          clrscr();
          printf("\nEnter the number of blocks:");
          scanf("%d",&nb);
          printf("Enter the number of files:");
          scanf("%d",&nf);
          printf("\nEnter the size of the blocks:-\n");
          for(i=1;i<=nb;i++)
      printf("Block %d:",i);
      scanf("%d",&b[i]);
         printf("Enter the size of the files :-\n");
         for(i=1;i<=nf;i++)
         {
                  printf("File %d:",i);
                  scanf("%d",&f[i]);
         }
          for(i=1;i<=nf;i++)
         {
                   for(j=1;j<=nb;j++)
                  {
                           if(bf[j]!=1)
                           {
                                    temp=b[j]-f[i];
                                    if(temp>=0)
                                             if(lowest>temp)
                                             {
                                             ff[i]=j;
                                              lowest=temp;
                                              }
                        }}
                   frag[i]=lowest;  bf[ff[i]]=1;  lowest=10000;
         }
          printf("\nFile No\tFile Size \tBlock No\tBlock
          Size\tFragment"); for(i=1;i<=nf && ff[i]!=0;i++)

          printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
          getch();
}
```

Output :

```
Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:-
Block 1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:-
File 1: 1
File 2: 4
```

	File No	File Size	Block No	Block Size	Fragment
1	1		2	2	1
2	4		1	5	1

WORST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
        int frag[max],b[max],f[max],i,j,nb,nf,temp;
        static int bf[max],ff[max];
        clrscr();
        printf("\n\tMemory Management Scheme - First Fit");
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
        for(i=1;i<=nb;i++)
        {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
        }
        printf("Enter the size of the files :-\n");
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
```

```
        }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1)
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                {
                                        ff[i]=j;
                                        break;
                                }
                        }
                }
                frag[i]=temp;
                bf[ff[i]]=1;
        }
        printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
        for(i=1;i<=nf;i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        getch();
}
```

Output :

```
Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:-
Block 1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:-
File 1: 1
File 2: 4
```

File No	File Size	Block No	Block Size	Fragment
1	1	1	5	4
2	4	3	7	3

4. Simulate Bankers Algorithm for Dead Lock Avoidance & Prevention.

```
#include<stdio.h>
#include<conio.h>
struct da
{
int max[10],a1[10],need[10],before[10],after[10];
}p[10];
void main()
{
int i,j,k,l,r,n,tot[10],av[10],cn=0,cz=0,temp=0,c=0;
clrscr();
printf("\n ENTER THE NO. OF PROCESSES:");
scanf("%d",&n);
printf("\n ENTER THE NO. OF RESOURCES:");
scanf("%d",&r);
for(i=0;i<n;i++)
{
printf("PROCESS %d \n",i+1);
for(j=0;j<r;j++)
{
printf("MAXIMUM VALUE FOR RESOURCE %d:",j+1);
scanf("%d",&p[i].max[j]);
}
for(j=0;j<r;j++)
{
printf("ALLOCATED FROM RESOURCE %d:",j+1);
scanf("%d",&p[i].a1[j]);
p[i].need[j]=p[i].max[j]-p[i].a1[j];
}
}
for(i=0;i<r;i++)
{
printf("ENTER TOTAL VALUE OF RESOURCE %d:",i+1);
scanf("%d",&tot[i]);
}
for(i=0;i<r;i++)
{
for(j=0;j<n;j++)
temp=temp+p[j].a1[i];
av[i]=tot[i]-temp;
temp=0;
}
printf("\n\t RESOURCES ALLOCATED   NEEDED   TOTAL AVAIL");
```

```
for(i=0;i<n;i++)
{
printf("\n P%d \t",i+1);
for(j=0;j<r;j++)
printf("%d",p[i].max[j]);
printf("\t");
for(j=0;j<r;j++)
printf("%d",p[i].a1[j]);
printf("\t");
for(j=0;j<r;j++)
printf("%d",p[i].need[j]);
printf("\t");
for(j=0;j<r;j++)
{
if(i==0)
printf("%d",tot[j]);
}
printf("         ");
for(j=0;j<r;j++)
{
if(i==0)
printf("%d",av[j]);
}
}
printf("\n\n\t AVAIL BEFORE\T AVAIL AFTER ");
for(l=0;l<n;l++)
{
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
if(p[i].need[j] >av[j])
cn++;
if(p[i].max[j]==0)
cz++;
}
if(cn==0 && cz!=r)
{
for(j=0;j<r;j++)
{
p[i].before[j]=av[j]-p[i].need[j];
p[i].after[j]=p[i].before[j]+p[i].max[j];
av[j]=p[i].after[j];
p[i].max[j]=0;
}
printf("\n P %d \t",i+1);
for(j=0;j<r;j++)
```

```
printf("%d",p[i].before[j]);
printf("\t");
for(j=0;j<r;j++)
printf("%d",p[i].after[j]);
cn=0;
cz=0;
c++;
break;
}
else
{
cn=0;
cz=0;
}
}
}
if(c==n)
printf("\n THE ABOVE SEQUENCE IS A SAFE SEQUENCE");
else
printf("\n DEADLOCK OCCURED");
getch();
}
```

Output :

ENTER THE NO. OF PROCESSES:4

ENTER THE NO. OF RESOURCES:3
PROCESS 1
MAXIMUM VALUE FOR RESOURCE 1:3
MAXIMUM VALUE FOR RESOURCE 2:2
MAXIMUM VALUE FOR RESOURCE 3:2

ALLOCATED FROM RESOURCE 1:1
ALLOCATED FROM RESOURCE 2:0
ALLOCATED FROM RESOURCE 3:0

PROCESS 2
MAXIMUM VALUE FOR RESOURCE 1:6
MAXIMUM VALUE FOR RESOURCE 2:1
MAXIMUM VALUE FOR RESOURCE 3:3

ALLOCATED FROM RESOURCE 1:5
ALLOCATED FROM RESOURCE 2:1
ALLOCATED FROM RESOURCE 3:1

PROCESS 3
MAXIMUM VALUE FOR RESOURCE 1:3
MAXIMUM VALUE FOR RESOURCE 2:1
MAXIMUM VALUE FOR RESOURCE 3:4

ALLOCATED FROM RESOURCE 1:2
ALLOCATED FROM RESOURCE 2:1
ALLOCATED FROM RESOURCE 3:1

PROCESS 4
MAXIMUM VALUE FOR RESOURCE 1:4
MAXIMUM VALUE FOR RESOURCE 2:2
MAXIMUM VALUE FOR RESOURCE 3:2

ALLOCATED FROM RESOURCE 1:0
ALLOCATED FROM RESOURCE 2:0
ALLOCATED FROM RESOURCE 3:2

ENTER TOTAL VALUE OF RESOURCE 1:9
ENTER TOTAL VALUE OF RESOURCE 2:3

ENTER TOTAL VALUE OF RESOURCE 3:6

RESOURCES	ALLOCATED	NEEDED	TOTAL	AVAIL
P1	322	100	222	936 112
P2	613	511	102	
P3	314	211	103	
P4	422	002	420	

	AVAIL BEFORE	AVAIL AFTER
P 2	010	623
P 1	401	723
P 3	620	934
P 4	514	936

THE ABOVE SEQUENCE IS A SAFE SEQUENCE

5. Simulate The Following Page Replacement Algorithms.

a) FIFO

```
#include<stdio.h>
#include<conio.h>
int fr[3];
void main()
{
void display();
int i,j,page[12]={2,3,2,1,5,2,4,5,3,2,5,2};
int
flag1=0,flag2=0,pf=0,frsize=3,top=0;
clrscr();
for(i=0;i<3;i++)
{
fr[i]=-1;
}
for(j=0;j<12;j++)
{
flag1=0; flag2=0;
for(i=0;i<12;i++)
{
if(fr[i]==page[j])
{
flag1=1; flag2=1;
break;
}
}
if(flag1==0)
{
for(i=0;i<frsize;i++)
{
if(fr[i]==-1)
{
fr[i]=page[j]; flag2=1;
break;
}
}
}
if(flag2==0)
{
fr[top]=page[j];
top++;
pf++;
if(top>=frsize)
```

```
top=0;
}
display();
}
```

```
printf("Number of page faults : %d",pf+frsize);
 getch();
}
void display()
{
int     i;
printf("\n");
for(i=0;i<3;i++)
printf("%d\t",fr[i]);
}
```

OUTPUT:

```
2 -1 -1
2  3 -1
2  3 -1
2  3  1
5  3  1
5  2  1
5  2  4
5  2  4
3  2  4
3  2  4
3  5  4
3  5  2
```

Number of page faults: 9

b) LRU

```
#include<stdio.h>
#include<conio.h>
int fr[3];
void main()
{
void display();
int p[12]={2,3,2,1,5,2,4,5,3,2,5,2},i,j,fs[3];
int index,k,l,flag1=0,flag2=0,pf=0,frsize=3;
 clrscr();
for(i=0;i<3;i++)
{
fr[i]=-1;
}
for(j=0;j<12;j++)
{
flag1=0,flag2=0;
for(i=0;i<3;i++)
{
if(fr[i]==p[j])
{
flag1=1;
flag2=1; break;
}
}
if(flag1==0)
```

```
{
for(i=0;i<3;i++)
{
if(fr[i]==-1)
{
fr[i]=p[j];
flag2=1;
break;
}
}
}
if(flag2==0)
{
for(i=0;i<3;i++)
fs[i]=0;
for(k=j-1,l=1;l<=frsize-1;l++,k--)
{
for(i=0;i<3;i++)
{
if(fr[i]==p[k])
fs[i]=1;
}}
for(i=0;i<3;i++)
{
if(fs[i]==0)
index=i;
}
fr[index]=p[j];
pf++;
}
display();
}
printf("\n no of page faults :%d",pf+frsize);
getch();
}
void display()
{
int i; printf("\n");
for(i=0;i<3;i++)
printf("\t%d",fr[i]);
}
```

OUTPUT:

2 -1 -1
2 3 -1
2 3 -1
2 3 1
2 5 1
2 5 1
2 5 4
2 5 4
3 5 4
3 5 2
3 5 2
3 5 2

No of page faults: 7

c) LFU

```
#include<stdio.h>
#include<conio.h>
int fr[3], n, m;
void
display();
void main()
{
 int i,j,page[20],fs[10];
 int
 max,found=0,lg[3],index,k,l,flag1=0,flag2=0,pf=0;
 float pr;
 clrscr();
 printf("Enter length of the reference string: ");
 scanf("%d",&n);
 printf("Enter the reference string: ");
 for(i=0;i<n;i++)
 scanf("%d",&page[i]);
 printf("Enter no of frames: ");
 scanf("%d",&m);
 for(i=0;i<m;i++)
 fr[i]=-1; pf=m;
```

```
for(j=0;j<n;j++)
 {
 flag1=0;
 flag2=0;
 for(i=0;i<m;i++)
 {
 if(fr[i]==page[j])
 {
 flag1=1;
 flag2=1;
 break;
 }
 }
 if(flag1==0)
 {
 for(i=0;i<m;i++)
 {
 if(fr[i]==-1)
 {
fr[i]=page[j];
flag2=1;
break;
 }
 }
 }
 if(flag2==0)
 {
 for(i=0;i<m;i++)
 lg[i]=0;
 for(i=0;i<m;i++)
 {
 for(k=j+1;k<=n;k++)
 {
 if(fr[i]==page[k])
 {
 lg[i]=k-j;
 break;
 }
 }
 }
 found=0;
 for(i=0;i<m;i++)
 {
 if(lg[i]==0)
 {
index=i;
```

found =1;

```
 break;
 }
 }
if(found==0)
 {
 max=lg[0];
 index=0;
 for(i=0;i<m;i++)
 {
 if(max<lg[i])
{
 max=lg[i];
 index=i;
}
 }
 }
 fr[index]=page[j];
 pf++;
}
display();
}
printf("Number of page faults : %d\n", pf);
pr=(float)pf/n*100;
printf("Page fault rate = %f \n", pr);
getch();
}
void display()
{
 int i;
 for(i=0;i<m;i++)
 printf("%d\t",fr[i]);
 printf("\n");
}
```

OUTPUT:

```
Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 3
1 -1 -1
1 2 -1
1 2 3
1 2 4
1 2 4
1 2 4
1 2 5
1 2 5
1 2 5
3 2 5
4 2 5
4 2 5
Number of page faults : 7 Page fault rate = 58.333332
```

6. Simulate the Following File Allocation Strategies
a) Sequenced

```
#include<stdio.h>
main()
{
int f[50],i,st,j,len,c,k;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
X: printf("\n Enter the starting block & length of file");
scanf("%d%d",&st,&len);
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("\n%d->%d",j,f[j]);
}
else
{
printf("Block already allocated");
break;
}
if(j==(st+len))
printf("\n the file is allocated to disk");
printf("\n if u want to enter more files?(y-1/n-0)");
scanf("%d",&c);
if(c==1)
goto X;
else
exit();
getch();
}
```

Output:

Enter the starting block & length of file 4 10
4->1
5->1
6->1
7->1
8->1
9->1
10->1
11->1
12->1
13->1
The file is allocated to disk.

b) Indexed

```
#include<stdio.h>
int    f[50],i,k,j,inde[50],n,c,count=0,p;
main()
{
clrscr();
for(i=0;i<50;i++)
f[i]=0;
x: printf("enter index block\t");
scanf("%d",&p);
if(f[p]==0)
{
f[p]=1;
printf("enter no of files on index\t");
scanf("%d",&n);
}
else
{
printf("Block already allocated\n");
goto x;
}
for(i=0;i<n;i++)
scanf("%d",&inde[i]);
for(i=0;i<n;i++)
if(f[inde[i]]==1)
{
printf("Block already allocated");
goto x;
}
for(j=0;j<n;j++)
f[inde[j]]=1;
printf("\n       allocated");
printf("\n file indexed");
for(k=0;k<n;k++)
printf("\n %d->%d:%d",p,inde[k],f[inde[k]]);
printf(" Enter 1 to enter more files and 0 to exit\t");
scanf("%d",&c);
if(c==1)
goto x;
else
exit();
getch();
}
```

Output :

enter index block 9

enter no of files on index 3

12 3

Allocated

File indexed

9->1:1

9->2;1

9->3:1

enter 1 to enter more files and 0 to exit

c) Linked

```
#include<stdio.h>
main()
{
int f[50],p,i,j,k,a,st,len,n,c;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Enter how many blocks that are alreadyallocated");
scanf("%d",&p);
printf("\nEnter the blocks no.s that are already allocated");
for(i=0;i<p;i++)
{
scanf("%d",&a);
f[a]=1;
}
X: printf("Enter  the starting index block &length");
scanf("%d%d",&st,&len);
k=len;
for(j=st;j<(k+st);j++)
{
if(f[j]==0)
{
f[j]=1;
printf("\n%d->%d",j,f[j]);
}
else
{
printf("\n %d->file is alreadyallocated",j);
```

```
k++;
}
}
 printf("\n If u want to enter onemore file? (yes-1/no-0)");
scanf("%d",&c);
if(c==1)
 gotoX;
else
exit();
getch( );}
```

Output :

Enter how many blocks that are already allocated 3
Enter the blocks no.sthat are already allocated 4 7
Enter the starting index block & length 3 7 9
3->1
4->1 file is already allocated
5->1
6->1
7->1 file is already allocated
8->1
9->1file is already
allocated10->1
11->1
12->1

****