6. Write C programs to simulate the following memory management techniques
a) Paging b) Segmentation

a) Paging

```
#include<stdio.h>
int main()
{
 int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;
 int s[10], fno[10][20];
 printf("\nEnter the memory size -- ");
 scanf("%d",&ms);
 printf("\nEnter the page size -- ");
 scanf("%d",&ps);
 nop = ms/ps;
 printf("\nThe no. of pages available in memory are -- %d ",nop);
 printf("\nEnter number of processes -- ");
 scanf("%d",&np);
 rempages = nop;
 for(i=1;i<=np;i++)
 {
  printf("\nEnter no. of pages required for p[%d]-- ",i);
  scanf("%d",&s[i]);
   if(s[i] >rempages)
   {       printf("\nMemory is Full");
          break;
   }
 rempages = rempages - s[i];
 printf("\nEnter pagetable for p[%d] --- ",i);
 for(j=0;j<s[i];j++)
  scanf("%d",&fno[i][j]);
 }
  printf("\nEnter Logical Address to find Physical Address ");
  printf("\nEnter process no. and pagenumber and offset -- ");
  scanf("%d %d %d",&x,&y, &offset);
   if(x>np || y>=s[x] || offset>=ps)
   printf("\nInvalid Process or Page Number or offset");
   else
   {     pa=fno[x][y]*ps+offset;
         printf("\nThe Physical Address is -- %d",pa);
    }
     return 0;
```

}

OUTPUT 1:

Enter the memory size – 1000
Enter the page size – 100
The no. of pages available in memory are – 10
Enter number of processes – 3
Enter no. of pages required for p[1]—4
Enter pagetable for p[1] --- 8 6 9 5
Enter no. of pages required for p[2]—5
Enter pagetable for p[2] --- 1 4 5 7 3
Enter no. of pages required for p[3]—5
Memory is Full
Enter Logical Address to find Physical Address
Enter process no. and pagenumber and offset -- 2 3 60
The Physical Address is – 760

OUTPUT 2:

Enter the memory size – 1000
Enter the page size – 100
The no. of pages available in memory are – 10
Enter number of processes – 2
Enter no. of pages required for p[1]—2
Enter pagetable for p[1] --- 5 4
Enter no. of pages required for p[2]—3
Enter pagetable for p[2] --- 2 3 6
Enter Logical Address to find Physical Address
Enter process no. and pagenumber and offset -- 1 1 20
The Physical Address is – 420

b) Segmentation

```
#include<stdio.h>
#include<unistd.h>
```

```
#include<stdlib.h>
int main()
{
int b[20],l[20],n,i,pa,s,d;
printf("\nEnter the number of segments:");
scanf("%d",&n);
printf("\nEnter the base address and limit register:");
for(i=1;i<=n;i++)
{
scanf("%d",&b[i]);
scanf("%d",&l[i]);
}
 printf("\nEnter the logical address:");
 printf("\nEnter the segment number and the offset");
 scanf("%d %d",&s,&d);
 for(i=1;i<=n;i++)
 {
  if(i==s)
  {
   if(d<l[i])
   { pa=b[i]+d;
     printf("\nSegment No.\t Physical Address. \n %d \t \t %d \n",s,pa);
    exit(0);
    }
   else
   {    printf("\nsize of the segment exceeds");
       exit(0);
    }
  }
 }
printf("\nInvalid segment");
return 0;
}
```

OUTPUT 1:

```
Enter the number of segments: 3
Enter the base address and limit register:
```

100 50
150 20
130 34
Enter the logical address:
Enter the segment number and the offset :1 25
Segment No. Physical Address
1 125

OUTPUT 2:

Enter the number of segments: 3
Enter the base address and limit register:
100 50
150 20
130 34
Enter the logical address:
Enter the segment number and the offset :4 50
Invalid Segment

OUTPUT 3:

Enter the number of segments: 3
Enter the base address and limit register:
100 50
150 20
130 34
Enter the logical address:
Enter the segment number and the offset :1 60
size of the segment exceeds