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

// a) Paging

Algorithm:

Step1: Start the program.

Step2: Read the base address, page size, number of pages and memory unit.

Step3: If the memory limit is less than the base address display the memory limit is less than limit.

Step4: Create the page table with the number of pages and page address.

Step5: Read the page number and displacement value.

Step6: If the page number and displacement value is valid, add the displacement value with the address

corresponding to the page number and display the result.

Step7: Display the page is not found or displacement should be less than page size.

Step8: Stop the program

Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;
 int s[10], fno[10][20];

//clrscr();

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[i] || offset>=ps)
printf("\nInvalid Process or Page Number or offset");

else
{ pa=fno[x][y]*ps+offset;
printf("\nThe Physical Address is -- %d",pa);

}
getch();
}
```

Output:

```
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

b) Segmentation

Step1: Start the program.

Step2: Read the base address, number of segments, size of each segment, memory limit.

Step3: If memory address is less than the base address display “invalid memory limit”.

Step4: Create the segment table with the segment number and segment address and display it.

Step5: Read the segment number and displacement.

Step6: If the segment number and displacement is valid compute the real address and display the same.

Step7: Stop the program

Code:

```
#include <stdio.h>
#include <math.h>
int sost;
void gstinfo();
void ptladdr();
struct segtab
{
```

```
 int sno;
 int baddr;
 int limit;
 int val[10];
}st[10];
void gstinfo()
{
 int i,j;
 printf("\n\tEnter the size of the segment table: ");
 scanf("%d",&sost);
 for(i=1;i<=sost;i++)
 {
 printf("\n\tEnter the information about segment: %d",i);
 st[i].sno = i;
 printf("\n\tEnter the base Address: ");
 scanf("%d",&st[i].baddr);
 printf("\n\tEnter the Limit: ");
 scanf("%d",&st[i].limit);
 for(j=0;j<st[i].limit;j++)
 {
 printf("Enter the %d address Value: ",(st[i].baddr + j));
 scanf("%d",&st[i].val[j]);
 }
 }
}
void ptladdr()
{
 int i,swd,d=0,n,s,disp,paddr;
```

```
//clrscr();
printf("\n\n\t\t\t SEGMENT TABLE \n\n");
printf("\n\t\t SEG.NO\t\tBASE ADDRESS\t\t LIMIT \n\n");
for(i=1;i<=sost;i++)
printf("\t\t%d \t\t%d\t\t%d\n\n",st[i].sno,st[i].baddr,st[i].limit);
printf("\n\nEnter the logical Address: ");
scanf("%d",&swd);
n=swd;
while (n != 0)
{
n=n/10;
d++;
}
s = swd/pow(10,d-1);
disp = swd%(int)pow(10,d-1);
if(s<=sost)
{
if(disp < st[s].limit)
{
paddr = st[s].baddr + disp;
printf("\n\t\tLogical Address is: %d",swd);
printf("\n\t\tMapped Physical address is: %d",paddr);
printf("\n\tThe value is: %d",( st[s].val[disp] ) );
}
else
printf("\n\t\tLimit of segment %d is high\n\n",s);
}
else
```

```
printf("\n\t\tInvalid Segment Address \n");
}
void main()
{
char ch;
//clrscr();
gstinfo();
do
{
ptladdr();
printf("\n\t Do U want to Continue(Y/N)");
//flushall();
scanf("%c",&ch);
}while (ch == 'Y' || ch == 'y' );
//getch();
}
```

Output:

```
Enter the size of the segment table: 3

    Enter the information about segment: 1
    Enter the base Address: 3

    Enter the Limit: 5
Enter the 3 address Value: 1
Enter the 4 address Value: 2
Enter the 5 address Value: 3
Enter the 6 address Value: 4
```

Enter the 7 address Value: 5

Enter the information about segment: 2

Enter the base Address: 6

Enter the Limit: 4

Enter the 6 address Value: 11

Enter the 7 address Value: 12

Enter the 8 address Value: 13

Enter the 9 address Value: 14

Enter the information about segment: 3

Enter the base Address: 9

Enter the Limit: 5

Enter the 9 address Value: 22

Enter the 10 address Value: 23

Enter the 11 address Value: 24

Enter the 12 address Value: 25

Enter the 13 address Value: 26

SEGMENT TABLE

SEG.NO	BASE ADDRESS	LIMIT
1	3	5

2 6 4

3 9 5

Enter the logical Address: 3

Logical Address is: 3

Mapped Physical address is: 9

The value is: 22

Do U want to Continue(Y/N)

Process exited after 76.35 seconds with return value 10

Press any key to continue . . .