Week-9:

Write a C program to simulate paging technique of memory management. (create a logical memory space, physical memory space and page table, you should show the address translation entirely)

Code:

```
#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 and page size -- ");
   scanf("%d %d",&ms,&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);
  }
  return 0;
}
```

Output:

```
Enter the memory size and page size -- 1000 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
```