/*Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method), and implement hashing
technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.
*/

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define BS 10
typedef struct employee
{
   int enum;
   char name[20];
}EMP;

EMP HT[BS], input;

FILE *fp;


void disp()
{
   int i;
   printf("\n\n***********HASH TABLE *******\n\n");
   printf("Bucketno\t EMP NUM\t NAME\n");
   for(i=0;i<BS;i++)
      printf("%d\t\t %d\t\t %s\n",i,HT[i].enum, HT[i].name);
}



void main()
{
int i,key,j,c=0;

fp=fopen("EMP.dat","w+");

for(i=0;i<BS;i++)
{
   HT[i].enum=-1;
   strcpy(HT[i]. name," ");
}
```

```
printf("enter 0 for enum to end input\n");

while(1)
{
   printf("\n emp num:");
   scanf("%d",&input.enum);

   if(input.enum==0)
      break;

   printf("first name:");
   scanf("%s",input.name);

   fwrite(&input, sizeof(struct employee),1,fp);
}
fclose(fp);

fp=fopen("EMP.dat","r");
while(fread(&input,sizeof(struct employee),1,fp ))
{
   c++;
   if(c>BS)
   {
      printf("\n\n**********HASH TABLE OVERFLOW*******\n\n");
      break;
   }

    key=input.enum%BS;

   if(HT[key].enum==-1)
   {
      HT[key].enum=input.enum;
      strcpy(HT[key].name,input.name);

    }
else
  {
     // printf("\n *********COLLISION AT %d bucket for emp
number=%d*****\n",key,input.enum);
   j=(key+1)%BS;
   while(j!=key)
   {
     if(HT[j].enum==-1)
      {
     // printf("\n*************USING LINEAR PROBING TO HANDLE COLLISION****\n");
     // printf("\n bucket %d is allocated for key %d\n",j,key);
      HT[j].enum=input.enum;
      strcpy(HT[j].name,input.name);
```

```
      break;
      }
     else
      j=(j+1)%BS;
    }
  }
 }
  disp();
}
```