```
/*12. 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.
  Design and 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>

#define SIZE 100
struct Employee
{
      int employee_key, employee_age;
      char employee_name[30];
};
struct Employee emp[SIZE];
int L;

int hash_function(int key)
{
    return (key % 97);
}

void linear_probing(int loc, struct Employee e)
{
    int count = 0, i, flag=0;
    printf("\nCollision Detected...!!!\n");
    i=0;
    while(i<SIZE)
    {
        if (emp[i].employee_key!=-1)
        count++;
        i++;
    }
    printf("Collision avoided successfully\n");
    if(count == SIZE)
    {
        printf("\n Hashing table is full\n");
        exit(1);
    }
    for(i=loc+1; i<SIZE; i++)
        if(emp[i].employee_key == -1)
        {
            emp[i] = e;
            flag =1;
            break;
        }
    i=0;
    while((i<loc) && (flag==0))
    {
        if(emp[i].employee_key == -1)
        {
            emp[i] = e;
            flag =1;
            break;
        }
        i++;
    }
}

void read_file()
{
 FILE *infile;
    int i, j;
    struct Employee E;
    infile = fopen("input.txt", "r");
    if(infile == NULL)
    {
        fprintf(stderr, "\nError opening input.txt\n\n");
        exit (1);
    }
    else
    {
        for (i=0; i<SIZE; i++)
        {
            emp[i].employee_key = -1;
        }
        while(fscanf(infile,"%d %s %d",&E.employee_key, E.employee_name,
&E.employee_age)!=EOF)
        {
            L=hash_function(E.employee_key);
            if(emp[L].employee_key == -1 )
                emp[L] = E;
            else
                linear_probing(L, E);
        }
    }
}

void display_attached()
{
    int j;
    printf("\n\n******Hash Table for Attached Keys*****\n\n");
        for (j=0; j<SIZE; j++)
        {
            if(emp[j].employee_key != -1)
            printf("LOCATION[%d]--->KEY= %d \t EMP NAME = %s \t EMP AGE =
%d\n",j,emp[j].employee_key, emp[j].employee_name, emp[j].employee_age);

        }
}

void display_all()
{
    int j;
    printf("\n\n*******Hash Table*******\n\n");
        for (j=0; j<SIZE; j++)
        {
            if(emp[j].employee_key != -1)
```

```
                printf("LOCATION[%d]--->KEY= %d \t EMP NAME = %s \t EMP
AGE = %d\n",j,emp[j].employee_key, emp[j].employee_name,
emp[j].employee_age);
            else
                printf("LOCATION[%d]--->EMPTY\n",j);

        }
}

void main()
{
    int ch;
    while (1)
    {
        printf("\n\n\n********** Hash function H: K L as H(K)=K mod m
(remainder method)**********\n\n");
        printf("1. Read Records from the Input file\n");
           printf("2. Display Key Attached Location of Hash Table\n");
           printf("3. Display All Location of Hash Table\n");
           printf("4. Exit\n");

           printf("Enter your choice:\n");
           scanf("%d", &ch);
           switch (ch)
           {
               case 1: read_file();
                    break;
               case 2: display_attached();
                    break;
            case 3: display_all();
                    break;
            case 4: exit(0);
            default: printf("Enter a Valid Choice\n");
                    break;
           }
    }
}
```