Program 12 : Hashing & Linear Probing

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 MAX_NUM_EMPLOYEES 100  // Maximum number of employees
#define MAX_HASH_TABLE_SIZE 50  // Maximum size of the hash table
// Define the structure for an employee record
typedef struct
{
   int iKey;  // 4-digit key
   char cName[50];
}EMPLOYEE;
// Define the hash table as an array of employee pointers
EMPLOYEE* stHashTable[MAX_HASH_TABLE_SIZE];
int fnCompHash(int, int);
void fnInsRecord(EMPLOYEE*, int);
EMPLOYEE* fnSrchRecord(int, int);
int main()
{
   int m;  // Size of the hash table
   printf("Enter the size of the hash table (m): ");
   scanf("%d",&m);
   // Initialize the hash table with NULL pointers
   for (int i = 0; i < m; i++)
         {
      stHashTable[i] = NULL;
   }
   FILE* file = fopen("employee.txt", "r");
   if(file == NULL)
         {
      printf("Error opening file.n");
      return 1;
   }
```

```
    int n = 0;
    EMPLOYEE emp;
    while(fscanf(file, "%d %s", &emp.iKey, emp.cName) != EOF)
        {
        EMPLOYEE* newEmp = (EMPLOYEE*)malloc(sizeof(EMPLOYEE));
        newEmp->iKey = emp.iKey;
        strcpy(newEmp->cName, emp.cName);
        fnInsRecord(newEmp, m);
        n++;
    }
    fclose(file);
    int iSrchKey;
    printf("Enter a key to search for an employee record: ");
    scanf("%d", &iSrchKey);
    EMPLOYEE* found = fnSrchRecord(iSrchKey, m);
    if(found != NULL)
        {
        printf("Employee found with key %d:\n", found->iKey);
        printf("Name: %s\n", found->cName);
    }
    else
        {
        printf("Employee with key %d not found.\n", iSrchKey);
    }
    return 0;
}
void fnInsRecord(EMPLOYEE* emp, int m)
{
    int index = fnCompHash(emp->iKey, m);
    // Linear probing if collisions happen
    while(stHashTable[index] != NULL)
        {
        index = (index + 1) % m;
    }
    stHashTable[index] = emp;
}
int fnCompHash(int iKey, int m)
{
    return iKey % m;
}
```

```
EMPLOYEE* fnSrchRecord(int iKey, int m)
{
   int index = fnCompHash(iKey, m);
   // Linear probing
   while(stHashTable[index] != NULL)
         {
      if(stHashTable[index]->iKey == iKey)
                  {
         return stHashTable[index];
      }
      index = (index + 1) % m;
   }
   return NULL; // Employee record not found
}
```

DATA File: save with the employee.txt

```
1414  ramdin
3671  sharat
4292  birender
2136  amit
6716  kushal
8458  kasid
9177  shiv
7521  vikram
5216  sanjay
2936  abhi
8590  raman
2963  khadak
6528  gurmit
1820  chanderpal
2387  aman
1796  khursid
9782  rajeev
4556  durgesh
4327  nahar
5263  ramesh
8758  sunder
8453  maansingh
8439  rohit
8539  john
```