Lab program 10:

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 MAX_EMPLOYEES 100

// Define the Hash Table size
#define m 100  // Number of memory locations in the hash table (L)

// Define the structure for an Employee Record
typedef struct {
    int key;  // The unique key for the employee (4-digit integer)
    int address;  // The address mapped in the hash table (2-digit integer)
} EmployeeRecord;

// Define the hash table
int hashTable[m];

// Hash function: H(K) = K mod m
int hashFunction(int key) {
    return key % m;
}

// Function to insert the key into the hash table using linear probing
int insert(int key) {
    int index = hashFunction(key);  // Compute hash index

    // If the location is empty, place the key there
    while (hashTable[index] != -1) {
        // If the location is already occupied, move to the next slot (linear probing)
        index = (index + 1) % m;
    }

    // Insert the key at the found index
    hashTable[index] = key;
```

```
    return index;
}

// Function to display the hash table
void displayHashTable() {
    printf("\nHash Table:\n");
    printf("Index  Key\n");
    for (int i = 0; i < m; i++) {
        if (hashTable[i] != -1) {
            printf("%d      %d\n", i, hashTable[i]);
        }
    }
}

int main() {
    // Initialize the hash table with -1 to indicate empty slots
    for (int i = 0; i < m; i++) {
        hashTable[i] = -1;
    }

    // Sample input: employee keys (4-digit integers)
    int employeeKeys[MAX_EMPLOYEES];
    int numEmployees;

    // Input number of employees
    printf("Enter number of employees: ");
    scanf("%d", &numEmployees);

    printf("Enter the employee keys (4-digit integers):\n");
    for (int i = 0; i < numEmployees; i++) {
        scanf("%d", &employeeKeys[i]);
    }

    // Insert each employee key into the hash table
    for (int i = 0; i < numEmployees; i++) {
        int address = insert(employeeKeys[i]);
        printf("Employee key %d inserted at address %d\n", employeeKeys[i], address);
    }

    // Display the final state of the hash table
    displayHashTable();

    return 0;
}
```

Output:

```
Enter number of employees: 5
Enter the employee keys (4-digit integers):
1234 5678 9101 1122 3344
Employee key 1234 inserted at address 34
Employee key 5678 inserted at address 78
Employee key 9101 inserted at address 1
Employee key 1122 inserted at address 22
Employee key 3344 inserted at address 44

Hash Table:
Index  Key
1      9101
22      1122
34      1234
44      3344
78      5678

Process returned 0 (0x0)   execution time : 29.792 s
Press any key to continue.
```