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

#define MAX_EMPLOYEES 100
#define TABLE_SIZE 10

// Structure to represent an employee record
struct Employee {
    int key;
    // Add other employee information as needed
    // For simplicity, only the key is considered here
};

// Structure to represent a hash table entry
struct HashEntry {
    struct Employee employee;
    int isOccupied;
};

// Function to initialize the hash table
void initializeHashTable(struct HashEntry hashTable[], int
size) {
    for (int i = 0; i < size; i++) {
        hashTable[i].isOccupied = 0; // Mark all entries as
unoccupied
    }
}

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

// Function to insert an employee record into the hash table
void insert(struct HashEntry hashTable[], int size, struct
Employee employee) {
    int key = employee.key;
    int index = hashFunction(key, size);

    // Linear probing to handle collisions
    while (hashTable[index].isOccupied) {
        index = (index + 1) % size; // Move to the next
position

        // If we looped back to the original position, the
table is full
        if (index == hashFunction(key, size)) {
            printf("Hash table is full. Cannot insert employee
with key %d.\n", key);
            return;
        }
    }

    // Insert the employee record at the computed index
    hashTable[index].employee = employee;
    hashTable[index].isOccupied = 1;
    printf("Employee with key %d inserted at index %d\n", key,
index);
}

// Function to display the contents of the hash table
void displayHashTable(struct HashEntry hashTable[], int size) {
    printf("\nHash Table:\n");
    printf("Index\tKey\n");

    for (int i = 0; i < size; i++) {
        printf("%d\t", i);

        if (hashTable[i].isOccupied) {
            printf("%d\n", hashTable[i].employee.key);
        } else {
            printf("Empty\n");
        }
    }
}

int main() {
    struct HashEntry hashTable[TABLE_SIZE];
    struct Employee employees[MAX_EMPLOYEES];

    int n; // Number of employee records
    printf("Enter the number of employee records: ");
    scanf("%d", &n);

    printf("Enter the employee records (key only):\n");
    for (int i = 0; i < n; i++) {
        printf("Employee %d key: ", i + 1);
        scanf("%d", &employees[i].key);
    }
    initializeHashTable(hashTable, TABLE_SIZE);

    // Insert employee records into the hash table
    for (int i = 0; i < n; i++) {
        insert(hashTable, TABLE_SIZE, employees[i]);
    }

    // Display the contents of the hash table
    displayHashTable(hashTable, TABLE_SIZE);

    return 0;
}
```