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

#define TABLE_SIZE 10

// Node for linked list in Chaining
struct Node {
    int key;
    struct Node* next;
};

// Hash Table with Chaining
struct HashTable {
    struct Node* table[TABLE_SIZE];
};

// Simple hash function
int hash(int key) {
    return key % TABLE_SIZE;
}

// Initialize hash table
void initHashTable(struct HashTable* ht) {
    for (int i = 0; i < TABLE_SIZE; i++) {
        ht->table[i] = NULL;
    }
}

// Insert into hash table (Chaining)
void insertChaining(struct HashTable* ht, int key) {
    int index = hash(key);
```

```
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->key = key;
    newNode->next = ht->table[index];
    ht->table[index] = newNode;
}

// Search in hash table (Chaining)
int searchChaining(struct HashTable* ht, int key) {
    int index = hash(key);
    struct Node* temp = ht->table[index];
    while (temp) {
        if (temp->key == key) return 1;
        temp = temp->next;
    }
    return 0;
}

// Print hash table (Chaining)
void printChaining(struct HashTable* ht) {
    for (int i = 0; i < TABLE_SIZE; i++) {
        printf("Bucket %d: ", i);
        struct Node* temp = ht->table[i];
        while (temp) {
            printf("%d -> ", temp->key);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

int main() {
```

```
    struct HashTable ht;
    initHashTable(&ht);
    insertChaining(&ht, 15);
    insertChaining(&ht, 25);
    insertChaining(&ht, 35);
    printChaining(&ht);
    printf("Found 25: %d\n", searchChaining(&ht, 25));
    return 0;
}
```