#include<stdio.h>

#define SIZE 10

typedef struct node {

int data;

int flag; // 0: empty, 1: occupied

} NODE;

void initializeHashTable(NODE \*hashTable) {

for (int i = 0; i < SIZE; i++) {

hashTable[i].data = 0;

hashTable[i].flag = 0;

}

}

// Insert using Linear Probing

void insert(int key, NODE \*hashTable) {

int hash, i = 0;

hash = ((key % SIZE) + i) % SIZE;

while (hashTable[hash].flag != 0 && i < SIZE) {

i++; // Increment number of collisions

hash = ((key % SIZE) + i) % SIZE; // New hash value

}

if (hashTable[hash].flag == 0) {

hashTable[hash].data = key;

hashTable[hash].flag = 1;

printf("The data %d is inserted at %d\n", key, hash);

} else {

printf("\nData cannot be inserted\n");

}

}

// Search using Linear Probing

void search(int key, NODE \*hashTable) {

int hash, i = 0;

hash = ((key % SIZE) + i) % SIZE;

while (i < SIZE) {

if (hashTable[hash].data == key && hashTable[hash].flag == 1) {

printf("\nThe data %d is found at location %d\n", key, hash);

return;

}

i++;

hash = ((key % SIZE) + i) % SIZE;

}

printf("\nData not found\n");

}

void display(NODE \*hashTable) {

printf("\nHash Table:\n");

for (int i = 0; i < SIZE; i++) {

if (hashTable[i].flag == 1) {

printf("Index %d: %d\n", i, hashTable[i].data);

} else {

printf("Index %d: EMPTY\n", i);

}

}

printf("\n");

}

int main() {

NODE hashTable[SIZE];

initializeHashTable(hashTable);

insert(5, hashTable);

insert(15, hashTable);

insert(25, hashTable);

display(hashTable);

search(15, hashTable); // Should find the data

search(35, hashTable); // Should not find the data

return 0;

}