#include <stdio.h>

#define SIZE 10

int hashTable[SIZE];

// Initialize hash table

void init() {

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

hashTable[i] = -1; // -1 indicates empty slot

}

// Hash function

int hash(int key) {

return key % SIZE;

}

// Insert using linear probing

void insert(int key) {

int index = hash(key);

int originalIndex = index;

while (hashTable[index] != -1) {

index = (index + 1) % SIZE;

if (index == originalIndex) {

printf("Hash table is full. Cannot insert %d\n", key);

return;

}

}

hashTable[index] = key;

printf("Inserted %d at index %d\n", key, index);

}

// Search using linear probing

void search(int key) {

int index = hash(key);

int originalIndex = index;

while (hashTable[index] != -1) {

if (hashTable[index] == key) {

printf("Found %d at index %d\n", key, index);

return;

}

index = (index + 1) % SIZE;

if (index == originalIndex) break;

}

printf("%d not found in the hash table\n", key);

}

// Display hash table

void display() {

printf("Hash Table:\n");

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

if (hashTable[i] != -1)

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

else

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

}

}

int main() {

int choice, key;

init();

while (1) {

printf("\nHashing with Linear Probing:\n");

printf("1. Insert\n2. Search\n3. Display\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter key to insert: ");

scanf("%d", &key);

insert(key);

break;

case 2:

printf("Enter key to search: ");

scanf("%d", &key);

search(key);

break;

case 3:

display();

break;

case 4:

return 0;

default:

printf("Invalid choice\n");

}

}

}