#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define TABLE\_SIZE 101

#define EMPTY -1

#define TOMBSTONE -2

typedef struct {

int key;

int value;

} HashEntry;

HashEntry hashTable[TABLE\_SIZE];

int hash(int key) {

return key % TABLE\_SIZE;

}

void initializeTable() {

for (int i = 0; i < TABLE\_SIZE; i++) {

hashTable[i].key = EMPTY;

hashTable[i].value = 0;

}

}

void insert(int key, int value) {

int index = hash(key);

int firstTombstone = -1;

for (int i = 0; i < TABLE\_SIZE; i++) {

int currentIndex = (index + i) % TABLE\_SIZE;

if (hashTable[currentIndex].key == EMPTY) {

if (firstTombstone != -1) {

currentIndex = firstTombstone;

}

hashTable[currentIndex].key = key;

hashTable[currentIndex].value = value;

return;

} else if (hashTable[currentIndex].key == TOMBSTONE) {

if (firstTombstone == -1) {

firstTombstone = currentIndex;

}

} else if (hashTable[currentIndex].key == key) {

hashTable[currentIndex].value = value;

return;

}

}

}

int search(int key) {

int index = hash(key);

for (int i = 0; i < TABLE\_SIZE; i++) {

int currentIndex = (index + i) % TABLE\_SIZE;

if (hashTable[currentIndex].key == EMPTY) {

return EMPTY;

} else if (hashTable[currentIndex].key == key) {

return hashTable[currentIndex].value;

}

}

return EMPTY;

}

void delete(int key) {

int index = hash(key);

for (int i = 0; i < TABLE\_SIZE; i++) {

int currentIndex = (index + i) % TABLE\_SIZE;

if (hashTable[currentIndex].key == EMPTY) {

return;

} else if (hashTable[currentIndex].key == key) {

hashTable[currentIndex].key = TOMBSTONE;

hashTable[currentIndex].value = 0;

return;

}

}

}

int main() {

initializeTable();

insert(1, 10);

insert(2, 20);

insert(102, 30); // Collision with key 1

printf("Value for key 1: %d\n", search(1));

printf("Value for key 2: %d\n", search(2));

printf("Value for key 102: %d\n", search(102));

delete(1);

printf("Value for key 1 after deletion: %d\n", search(1));

insert(203, 40); // Should recycle tombstone from key 1

printf("Value for key 203: %d\n", search(203));

return 0;

}