#include <stdio.h>

#include <stdlib.h>

#define MAX\_EMPLOYEES 100

#define EMPTY -1

typedef struct {

int key;

char name[50];

} Employee;

int m = 11;

int \*hashTable;

Employee employees[MAX\_EMPLOYEES];

int hashFunction(int key) {

return key % m;

}

void initializeHashTable() {

hashTable = (int\*) malloc(m \* sizeof(int));

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

hashTable[i] = EMPTY;

}

}

int linearProbe(int key) {

int index = hashFunction(key);

int startIdx = index;

while (hashTable[index] != EMPTY) {

index = (index + 1) % m;

if (index == startIdx) {

return -1;

}

}

return index;

}

void insertEmployee(int key, char\* name) {

int index = linearProbe(key);

if (index != -1) {

hashTable[index] = key;

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

} else {

printf("Error: Hash table is full, unable to insert key %d\n", key);

}

}

void displayHashTable() {

printf("Hash Table (memory locations):\n");

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

if (hashTable[i] != EMPTY) {

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

} else {

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

}

}

}

int main() {

initializeHashTable();

insertEmployee(1234, "John");

insertEmployee(5678, "Avinash");

insertEmployee(9101, "Amy");

insertEmployee(1122, "Balaji");

insertEmployee(5678, "Chandan");

displayHashTable();

free(hashTable);

return 0;

}

