Rajalakshmi Engineering College

Name: Sarvesh S
Email: 240701479@rajalakshmi.edu.in
Roll no: 240701479
Phone: 9361488694
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Develop a program using hashing to manage a fruit contest where each fruit is assigned a unique name and a corresponding score. The program should allow the organizer to input the number of fruits and their names with scores.

Then, it should enable them to check if a specific fruit, identified by its name, is part of the contest. If the fruit is registered, the program should display its score; otherwise, it should indicate that it is not included in the contest.

Input Format

The first line consists of an integer N, representing the number of fruits in the contest.

The following N lines contain a string K and an integer V, separated by a space, representing the name and score of each fruit in the contest.

The last line consists of a string T, representing the name of the fruit to search for.

Output Format

If T exists in the dictionary, print "Key "T" exists in the dictionary.".

If T does not exist in the dictionary, print "Key "T" does not exist in the dictionary.".

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 2
banana 2
apple 1
Banana
Output: Key "Banana" does not exist in the dictionary.

Answer

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

#define TABLE_SIZE 100

typedef struct {
    char key[100];
    int value;
    bool isOccupied;

} HashEntry;

HashEntry hashTable[TABLE_SIZE];
```

```
int hash(char *key){
    int sum = 0;
    for (int i = 0; key[i]; i++){
        sum += key[i];
    }
    return sum % TABLE_SIZE;
}

void insert(char *key, int value){
    int index = hash(key);
    int originalIndex = index;
    while (hashTable[index].isOccupied){
        if (strcmp(hashTable[index].key, key) == 0){
            hashTable[index].value = value;
            return;
        }
        index = (index + 1) % TABLE_SIZE;
        if (index == originalIndex){
            return;
        }
    }
    strcpy(hashTable[index].key, key);
    hashTable[index].value = value;
    hashTable[index].isOccupied = true;
}

int search(char *key) {
    int index = hash(key);
    int originalIndex = index;
    while (hashTable[index].isOccupied) {
        if (strcmp(hashTable[index].key, key) == 0){
            return 1;
        }
        index = (index + 1) % TABLE_SIZE;
        if (index == originalIndex) break;
    }
    return 0;
}

int main() {
    int n;
    scanf("%d", &n);
```

```
    for (int i = 0; i < TABLE_SIZE; i++){
        hashTable[i].isOccupied = false;
    }
    for (int i = 0; i < n; i++){
        char key[100];
        int value;
        scanf("%s %d", key, &value);
        insert(key, value);
    }
    char searchKey[100];
    scanf("%s", searchKey);
    if (search(searchKey)){
        printf("Key \"%s\" exists in the dictionary.\n", searchKey);
    }
    else{
        printf("Key \"%s\" does not exist in the dictionary.\n", searchKey);
    }
    return 0;
}
```

Status : Correct **Marks : 10/10**