Rajalakshmi Engineering College

Name: Harish S
Email: 240701173@rajalakshmi.edu.in
Roll no: 240701173
Phone: 9345569745
Branch: REC
Department: I CSE AG
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

```
// You are using GCC
// You are using GCC
#include <stdio.h>
#include <string.h>

#define MAX 30
#define TABLE_SIZE 31

typedef struct {
    char key[MAX];
    int value;
    int occupied;
} HashEntry;

int hashFunc(const char *key) {
    int hash = 0;
```

```
    for (int i = 0; key[i]; i++) {
        hash = (hash * 31 + key[i]) % TABLE_SIZE;
    }
    return hash;
}

void insert(HashEntry table[], const char *key, int value) {
    int idx = hashFunc(key);
    int start = idx;

    while (table[idx].occupied) {
        if (strcmp(table[idx].key, key) == 0) {
            table[idx].value = value;
            return;
        }
        idx = (idx + 1) % TABLE_SIZE;
        if (idx == start) return;
    }
    strcpy(table[idx].key, key);
    table[idx].value = value;
    table[idx].occupied = 1;
}

int search(HashEntry table[], const char *key) {
    int idx = hashFunc(key);
    int start = idx;

    while (table[idx].occupied) {
        if (strcmp(table[idx].key, key) == 0) {
            return table[idx].value;
        }
        idx = (idx + 1) % TABLE_SIZE;
        if (idx == start) break;
    }
    return -1;
}

int main() {
    int N;
    scanf("%d", &N);
    HashEntry table[TABLE_SIZE] = {0};
```

```
    for (int i = 0; i < N; i++) {
        char key[MAX];
        int val;
        scanf("%s %d", key, &val);
        insert(table, key, val);
    }

    char query[MAX];
    scanf("%s", query);

    int res = search(table, query);
    if (res == -1) {
        printf("Key \"%s\" does not exist in the dictionary.\n", query);
    } else {
        printf("Key \"%s\" exists in the dictionary.\n", query);
    }

    return 0;
}
```

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