Rajalakshmi Engineering College

Name: Sherwin G M
Email: 240701496@rajalakshmi.edu.in
Roll no: 240701496
Phone: 7708605966
Branch: REC
Department: I CSE FD
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
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 20

typedef struct {
    char key[50];
    int value;
    int isOccupied;
} FruitEntry;

FruitEntry hashTable[MAX_SIZE];
int size = 0;
```

```
int hashFunction(const char *key) {
    unsigned long hash = 0;
    while (*key) {
        hash = (hash * 31) + *key++;
    }
    return hash % MAX_SIZE;
}

void insertFruit(const char *key, int value) {
    int index = hashFunction(key);
    int originalIndex = index;
    while (hashTable[index].isOccupied) {
        if (strcmp(hashTable[index].key, key) == 0) {
            hashTable[index].value = value;
            return;
        }
        index = (index + 1) % MAX_SIZE;
        if (index == originalIndex) {
            return; // Table is full
        }
    }
    strcpy(hashTable[index].key, key);
    hashTable[index].value = value;
    hashTable[index].isOccupied = 1;
    size++;
}

int searchFruit(const char *key) {
    int index = hashFunction(key);
    int originalIndex = index;
    while (hashTable[index].isOccupied) {
        if (strcmp(hashTable[index].key, key) == 0) {
            return hashTable[index].value;
        }
        index = (index + 1) % MAX_SIZE;
        if (index == originalIndex) {
            break;
        }
    }
    return -1;
}
```

```
int main() {
    int N;
    scanf("%d", &N);
    char key[50];
    int value;
    for (int i = 0; i < N; i++) {
        scanf("%s %d", key, &value);
        insertFruit(key, value);
    }
    char T[50];
    scanf("%s", T);
    int result = searchFruit(T);
    if (result != -1) {
        printf("Key \"%s\" exists in the dictionary.\n", T);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", T);
    }
    return 0;
}
```

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