Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
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_FRUITS 15
#define HASH_TABLE_SIZE 31

typedef struct Fruit {
    char name[50];
    int score;
    struct Fruit *next;
} Fruit;

typedef struct HashTable {
    Fruit *table[HASH_TABLE_SIZE];
```

```
} HashTable;
unsigned int hash(const char *key) {
    unsigned int hashValue = 0;
    while (*key) {
        hashValue = (hashValue * 31 + *key) % HASH_TABLE_SIZE;
        key++;
    }
    return hashValue;
}
void initHashTable(HashTable *ht) {
    for (int i = 0; i < HASH_TABLE_SIZE; i++) {
        ht->table[i] = NULL;
    }
}

void insertFruit(HashTable *ht, const char *name, int score) {
    unsigned int index = hash(name);
    Fruit *newFruit = (Fruit *)malloc(sizeof(Fruit));
    strcpy(newFruit->name, name);
    newFruit->score = score;
    newFruit->next = ht->table[index];
    ht->table[index] = newFruit;
}

int searchFruit(HashTable *ht, const char *name, int *score) {
    unsigned int index = hash(name);
    Fruit *current = ht->table[index];
    while (current != NULL) {
        if (strcmp(current->name, name) == 0) {
            *score = current->score;
            return 1; // Found
        }
        current = current->next;
    }
    return 0;
}
void freeHashTable(HashTable *ht) {
    for (int i = 0; i < HASH_TABLE_SIZE; i++) {
        Fruit *current = ht->table[i];
        while (current != NULL) {
            Fruit *temp = current;
            current = current->next;
```

```
            free(temp);
        }
    }
}
int main() {
    HashTable ht;
    initHashTable(&ht);

    int N;
    scanf("%d", &N);

    char fruitName[50];
    int fruitScore;
    for (int i = 0; i < N; i++) {
        scanf("%s %d", fruitName, &fruitScore);
        insertFruit(&ht, fruitName, fruitScore);
    }
    scanf("%s", fruitName);
    int score;
    if (searchFruit(&ht, fruitName, &score)) {
        printf("Key \"%s\" exists in the dictionary.\n", fruitName);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", fruitName);
    }
    freeHashTable(&ht);
    return 0;
}
```

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