Rajalakshmi Engineering College

Name: Bebin Y
Email: 240801040@rajalakshmi.edu.in
Roll no: 240801040
Phone: 750858599
Branch: REC
Department: I ECE FA
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a messaging application, users maintain a contact list with names and corresponding phone numbers. Develop a program to manage this contact list using a dictionary implemented with hashing.

The program allows users to add contacts, delete contacts, and check if a specific contact exists. Additionally, it provides an option to print the contact list in the order of insertion.

Input Format

The first line consists of an integer n, representing the number of contact pairs to be inserted.

Each of the next n lines consists of two strings separated by a space: the name of the contact (key) and the corresponding phone number (value).

The last line contains a string k, representing the contact to be checked or removed.

Output Format

If the given contact exists in the dictionary:

1. The first line prints "The given key is removed!" after removing it.
2. The next n - 1 lines print the updated contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

If the given contact does not exist in the dictionary:

1. The first line prints "The given key is not found!".
2. The next n lines print the original contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

```
Input: 3
Alice 1234567890
Bob 9876543210
Charlie 4567890123
Bob
```

```
Output: The given key is removed!
Key: Alice; Value: 1234567890
Key: Charlie; Value: 4567890123
```

Answer

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

#define INITIAL_CAPACITY 10
```

```
typedef struct {
    char key[50];
    char value[50];
} KeyValuePair;

typedef struct {
    KeyValuePair *pairs;
    int size;
    int capacity;
} Dictionary;

void initDictionary(Dictionary *dict) {
    dict->size = 0;
    dict->capacity = INITIAL_CAPACITY;
    dict->pairs = (KeyValuePair *)malloc(dict->capacity * sizeof(KeyValuePair));
}

int hash(const char *key) {
    int hash = 0;
    for (int i = 0; key[i] != '\0'; i++) {
        hash += key[i];
    }
    return hash;
}

void insertKeyValuePair(Dictionary *dict, const char *key, const char *value) {
    if (dict->size == dict->capacity) {
        dict->capacity *= 2;
        dict->pairs = (KeyValuePair *)realloc(dict->pairs, dict->capacity *
sizeof(KeyValuePair));
    }
    strcpy(dict->pairs[dict->size].key, key);
    strcpy(dict->pairs[dict->size].value, value);
    dict->size++;
}

int doesKeyExist(Dictionary *dict, const char *key) {
    for (int i = 0; i < dict->size; i++) {
        if (strcmp(dict->pairs[i].key, key) == 0) {
            return i;
        }
    }
    return -1;
}

void removeKeyValuePair(Dictionary *dict, const char *key) {
    int index = doesKeyExist(dict, key);
    if (index != -1) {
        for (int i = index; i < dict->size - 1; i++) {
            strcpy(dict->pairs[i].key, dict->pairs[i + 1].key);
            strcpy(dict->pairs[i].value, dict->pairs[i + 1].value);
        }
        dict->size--;
        printf("The given key is removed!\n");
    }
}

void printDictionary(Dictionary *dict) {
    for (int i = 0; i < dict->size; i++) {
        printf("Key: %s; Value: %s\n", dict->pairs[i].key, dict->pairs[i].value);
    }
}

int main() {
    Dictionary dict;
    initDictionary(&dict);
    int numPairs;
    scanf("%d", &numPairs);
    char key[50], value[50];
    for (int i = 0; i < numPairs; i++) {
        scanf("%s %s", key, value);
        insertKeyValuePair(&dict, key, value);
    }
    scanf("%s", key);
    if (doesKeyExist(&dict, key) != -1) {
        removeKeyValuePair(&dict, key);
        printDictionary(&dict);
    } else {
        printf("The given key is not found!\n");
        printDictionary(&dict);
    }
    free(dict.pairs);
    return 0;
```

```
}
```

Status : Correct *Marks : 10/10*