# Rajalakshmi Engineering College

Name: Kavyasri M
Email: 240701248@rajalakshmi.edu.in
Roll no: 240701248
Phone: 6383586337
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 TABLE_SIZE 17
#define MAX_LEN 100
typedef struct Node {
   char key[MAX_LEN];
   int value;
   struct Node* next;
} Node;
Node* hashTable[TABLE_SIZE];
unsigned int hash(const char* str) {
   unsigned long hash = 5381;
   int c;
   while ((c = *str++))
```

```
        hash = ((hash << 5) + hash) + c;
    return hash % TABLE_SIZE;
}
void insert(const char* key, int value) {
    unsigned int index = hash(key);
    Node* newNode = (Node*)malloc(sizeof(Node));
    strcpy(newNode->key, key);
    newNode->value = value;
    newNode->next = hashTable[index];
    hashTable[index] = newNode;
}
Node* search(const char* key) {
    unsigned int index = hash(key);
    Node* current = hashTable[index];
    while (current != NULL) {
        if (strcmp(current->key, key) == 0)
            return current;
        current = current->next;
    }
    return NULL;
}
void freeTable() {
    for (int i = 0; i < TABLE_SIZE; ++i) {
        Node* current = hashTable[i];
        while (current != NULL) {
            Node* temp = current;
            current = current->next;
            free(temp);
        }
        hashTable[i] = NULL;
    }
}
int main() {
    int N;
    scanf("%d", &N);
    char key[MAX_LEN];
    int value;
    for (int i = 0; i < TABLE_SIZE; ++i)
        hashTable[i] = NULL;
    for (int i = 0; i < N; ++i) {
        scanf("%s %d", key, &value);
        insert(key, value);
    }
    char target[MAX_LEN];
    scanf("%s", target);
    Node* result = search(target);
    if (result != NULL) {
        printf("Key \"%s\" exists in the dictionary.\n", target);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", target);
    }
    freeTable();
    return 0;
}
```

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