Rajalakshmi Engineering College

Name: Vikashini M
Email: 241801314@rajalakshmi.edu.in
Roll no: 241801314
Phone: 9345747519
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TABLE_SIZE 101

typedef struct {
    char key[20];
    int value;
    int isOccupied;
} Entry;

Entry hashTable[TABLE_SIZE];
int hashFunction(const char *str) {
    int hash = 0;
    while (*str)
```

```
    hash = (hash * 31 + *str++) % TABLE_SIZE;
  return hash;
}
void insert(const char *key, int value) {
  int index = hashFunction(key);
  int originalIndex = index;

  while (hashTable[index].isOccupied && strcmp(hashTable[index].key, key) != 0)
{
    index = (index + 1) % TABLE_SIZE;
    if (index == originalIndex) return;
  }

  strcpy(hashTable[index].key, key);
  hashTable[index].value = value;
  hashTable[index].isOccupied = 1;
}


int search(const char *key, int *value) {
  int index = hashFunction(key);
  int originalIndex = index;

  while (hashTable[index].isOccupied) {
    if (strcmp(hashTable[index].key, key) == 0) {
      *value = hashTable[index].value;
      return 1;
    }
    index = (index + 1) % TABLE_SIZE;
    if (index == originalIndex) break;
  }
  return 0;
}

int main() {
  int n;
  scanf("%d", &n);

  char fruit[20];
  int score;

  for (int i = 0; i < n; i++) {
```

```
        scanf("%s %d", fruit, &score);
        insert(fruit, score);
    }

    char target[20];
    scanf("%s", target);

    int foundScore;
    if (search(target, &foundScore)) {
        printf("Key \"%s\" exists in the dictionary.\n", target);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", target);
    }

    return 0;
}
```

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