Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is building a basic hash table to manage student roll numbers for quick lookup. He decides to use Linear Probing to handle collisions.

Implement a hash table using linear probing where:

The hash function is: index = roll_number % table_sizeOn collision, check subsequent indexes (i+1, i+2, ...) until an empty slot is found.

You need to:

Insert a list of n student roll numbers into the hash table.Print the final state of the hash table. If a slot is empty, print -1.

Input Format

The first line of the input contains two integers n and table_size, where n is the

number of roll numbers to be inserted, and table_size is the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert into the hash table.

Output Format

The output should print a single line with table_size space-separated integers representing the final state of the hash table after all insertions.

If any slot remains unoccupied, it should be represented as -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4 7
50 700 76 85

Output: 700 50 85 -1 -1 -1 76

Answer

```
#include <stdio.h>

#define MAX 100

// You are using GCC
void initializeTable(int table[], int size) {
    for(int i=0;i<size;i++){
        table[i]=-1;
    }


}

int linearProbe(int table[], int size, int num) {
    int index=num%size;

    while(table[index]!=-1){
        index=(index+1)%size;
    }
```

```
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for(int j=0;j<n;j++){
        int index=linearProbe(table,size,arr[j]);
        table[index]=arr[j];
    }

}

void printTable(int table[], int size) {
    for(int k=0;k<size;k++){
        printf("%d\n",table[k]);
    }
}



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

    int arr[MAX];
    int table[MAX];

    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
    printTable(table, table_size);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Priya is developing a simple student management system. She wants to store roll numbers in a hash table using Linear Probing, and later search for specific roll numbers to check if they exist.

Implement a hash table using linear probing with the following operations:

Insert all roll numbers into the hash table.For a list of query roll numbers, print "Value x: Found" or "Value x: Not Found" depending on whether it exists in the table.

Input Format

The first line contains two integers, n and table_size — the number of roll numbers to insert and the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert.

The third line contains an integer q — the number of queries.

The fourth line contains q space-separated integers — the roll numbers to search for.

Output Format

The output print q lines — for each query value x, print: "Value x: Found" or "Value x: Not Found"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 10
21 31 41 51 61
3
31 60 51

Output: Value 31: Found
Value 60: Not Found
Value 51: Found

Answer

```
#include <stdio.h>

#define MAX 100


void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = 0;
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size;
    int start_index = index;
```

```
    while (table[index] != 0) {
        index = (index + 1) % size;
        if (index == start_index) {

            return -1;
        }
    }
    return index;
}


void insertIntoHashTable(int table[], int size, int arr[],int n) {
    for(int i=0;i<n;i++){
        int index=linearProbe(table,size,arr[i]);


    if (index != -1) {
        table[index] = arr[i];
    }
    }
}


int searchInHashTable(int table[], int size, int num) {
    int index = num % size;
    int start_index = index;

    while (table[index] != 0) {
        if (table[index] == num)
            return 1;
        index = (index + 1) % size;
        if (index == start_index)
            break;
    }
    return 0;
}


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

    int arr[MAX], table[MAX];
```

```
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);

    int q, x;
    scanf("%d", &q);
    for (int i = 0; i < q; i++) {
        scanf("%d", &x);
        if (searchInHashTable(table, table_size, x))
            printf("Value %d: Found\n", x);
        else
            printf("Value %d: Not Found\n", x);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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 TABLE_SIZE 101
```

```
typedef struct Node {
    char name[11];
    char phone[16];
    struct Node* next;
    struct Node* orderNext;
} Node;

Node* orderHead = NULL;
Node* orderTail = NULL;

Node* hashTable[TABLE_SIZE];


int hashFunction(const char* str) {
    int hash = 0;
    for (int i = 0; str[i]; i++) {
        hash = (hash * 31 + str[i]) % TABLE_SIZE;
    }
    return hash;
}

void insertContact(const char* name, const char* phone) {
    int index = hashFunction(name);


    Node* newNode = (Node*)malloc(sizeof(Node));
    strcpy(newNode->name, name);
    strcpy(newNode->phone, phone);
    newNode->next = hashTable[index];
    newNode->orderNext = NULL;


    hashTable[index] = newNode;

    if (!orderHead) {
        orderHead = orderTail = newNode;
    } else {
        orderTail->orderNext = newNode;
        orderTail = newNode;
    }
}
```

```
Node* searchContact(const char* name, Node** prev) {
    int index = hashFunction(name);
    Node* curr = hashTable[index];
    *prev = NULL;

    while (curr) {
        if (strcmp(curr->name, name) == 0) {
            return curr;
        }
        *prev = curr;
        curr = curr->next;
    }
    return NULL;
}

int removeContact(const char* name) {
    int index = hashFunction(name);
    Node* prev = NULL;
    Node* curr = searchContact(name, &prev);

    if (!curr) return 0;

    if (prev) {
        prev->next = curr->next;
    } else {
        hashTable[index] = curr->next;
    }
    Node* prevOrder = NULL;
    Node* currentOrder = orderHead;

    while (currentOrder) {
        if (strcmp(currentOrder->name, name) == 0) {
            if (prevOrder) {
                prevOrder->orderNext = currentOrder->orderNext;
            } else {
                orderHead = currentOrder->orderNext;
            }
            if (orderTail == currentOrder) {
                orderTail = prevOrder;
            }
            free(curr);
            return 1;
```

```
        }
        prevOrder = currentOrder;
        currentOrder = currentOrder->orderNext;
    }

    return 0;
}
void displayContacts() {
    Node* curr = orderHead;
    while (curr) {
        printf("Key: %s; Value: %s\n", curr->name, curr->phone);
        curr = curr->orderNext;
    }
}
int main() {
    int n;
    scanf("%d", &n);
    getchar();

    char name[11], phone[16];

    for (int i = 0; i < n; i++) {
        scanf("%s %s", name, phone);
        insertContact(name, phone);
    }

    char checkKey[11];
    scanf("%s", checkKey);

    if (removeContact(checkKey)) {
        printf("The given key is removed!\n");
        displayContacts();
    } else {
        printf("The given key is not found!\n");
        displayContacts();
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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

```
int keyExists(KeyValuePair* dictionary, int size, const char* key)
{
   for (int i = 0; i < size; i++)
{
      if (strcmp(dictionary[i].key, key) == 0)
{
         return 1;
}
}
   return 0;
}
```

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

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 0

Section 1 : MCQ

1. What is the output of the mid-square method for a key k = 123 if the hash table size is 10 and you extract the middle two digits of k * k?

Answer

Status : Skipped ***Marks : 0/1***

2. Which of the following best describes linear probing in hashing?

Answer

-

Status : - ***Marks : 0/1***

3. What does a deleted slot in linear probing typically contain?

Answer

-

Status : - *Marks : 0/1*

4. What is the initial position for a key k in a linear probing hash table?

Answer

-

Status : - *Marks : 0/1*

5. Which folding method divides the key into equal parts, reverses some of them, and then adds all parts?

Answer

-

Status : - *Marks : 0/1*

6. In the division method of hashing, the hash function is typically written as:

Answer

-

Status : - *Marks : 0/1*

7. Which of the following values of 'm' is recommended for the division method in hashing?

Answer

-

Status : - *Marks : 0/1*

8. What happens if we do not use modular arithmetic in linear probing?

Answer

-

Status : -

Marks : 0/1

9. What is the worst-case time complexity for inserting an element in a hash table with linear probing?

Answer

-

Status : -

Marks : 0/1

10. What is the primary disadvantage of linear probing?

Answer

-

Status : -

Marks : 0/1

11. Which situation causes clustering in linear probing?

Answer

-

Status : -

Marks : 0/1

12. What would be the result of folding 123456 into three parts and summing: (12 + 34 + 56)?

Answer

-

Status : -

Marks : 0/1

13. Which of these hashing methods may result in more uniform distribution with small keys?

Answer

-

Status : *-* ***Marks : 0/1***

14. In division method, if key = 125 and m = 13, what is the hash index?

Answer

-

Status : *-* ***Marks : 0/1***

15. Which of the following statements is TRUE regarding the folding method?

Answer

-

Status : *-* ***Marks : 0/1***

16. In the folding method, what is the primary reason for reversing alternate parts before addition?

Answer

-

Status : *-* ***Marks : 0/1***

17. Which C statement is correct for finding the next index in linear probing?

Answer

-

Status : - *Marks : 0/1*

18. In linear probing, if a collision occurs at index i, what is the next index checked?

Answer

-

Status : - *Marks : 0/1*

19. In C, how do you calculate the mid-square hash index for a key k, assuming we extract two middle digits and the table size is 100?

Answer

-

Status : - *Marks : 0/1*

20. Which data structure is primarily used in linear probing?

Answer

-

Status : - *Marks : 0/1*