Rajalakshmi Engineering College

Name: Nekhita Sri
Email: 241801184@rajalakshmi.edu.in
Roll no: 241801184
Phone: 8637459907
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

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 : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Nekhita Sri
Email: 241801184@rajalakshmi.edu.in
Roll no: 241801184
Phone: 8637459907
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

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

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

int linearProbe(int table[], int size, int num) {
    //Type your code here
```

```
    int index=num%size;
    int start=index;
    while(table[index]!=-1)
    {
        index=(index+1)%size;
        if(index==start)
        {
            return -1;
        }
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    //Type your code here
    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) {
    //Type your code here

    for(int i=0;i<size;i++)
    {
        if(table[i]==num)
        {
            return 1;
        }
    }
    return 0;
}

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

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

241801184

```
    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***

241801184

Rajalakshmi Engineering College

Name: Nekhita Sri
Email: 241801184@rajalakshmi.edu.in
Roll no: 241801184
Phone: 8637459907
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

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) {
    //Type your code here
    for(int i=0;i<size;i++)
    {
        table[i]=-1;
    }
}

int linearProbe(int table[], int size, int num) {
    //Type your code here
    int index=num%size;
    int start=index;
    while(table[index]!=-1)
    {
```

```
        index=(index+1)%size;
        if(index==start)
        {
            return -1;
        }
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    //Type your code here
    for(int i=0;i<n;i++)
    {
        int index=linearProbe(table,size,arr[i]);
        if(index!=-1)
        {
            table[index]=arr[i];
        }
    }
}

void printTable(int table[], int size) {
    //Type your code here
    for(int i=0;i<size;i++)
    {
        printf("%d ",table[i]);
    }
}
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);
```

241801184

```
    return 0;
}
```

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

241801184