Rajalakshmi Engineering College

Name: Divya Dharshini K
Email: 241501053@rajalakshmi.edu.in
Roll no: 241501053
Phone: 9566773627
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 17

Section 1 : MCQ

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

Answer

A special "deleted" marker

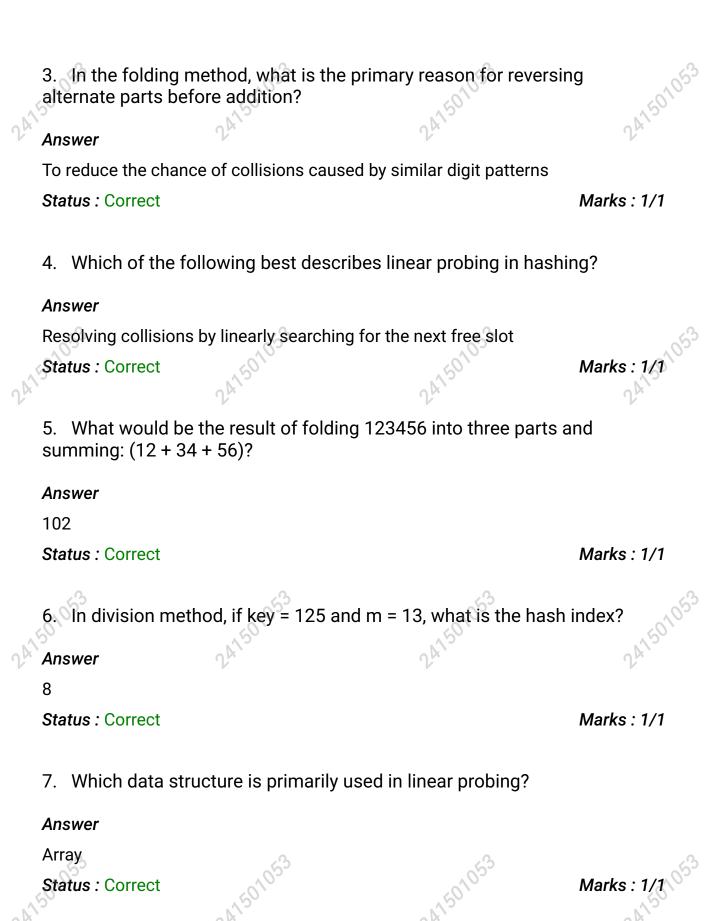*Status :* Correct   *Marks : 1/1*

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

Answer

k % table_size

Status : Correct   *Marks : 1/1*

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

Answer

To reduce the chance of collisions caused by similar digit patterns

Status : Correct   *Marks : 1/1*

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

Answer

Resolving collisions by linearly searching for the next free slot

Status : Correct   *Marks : 1/1*

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

Answer

102

Status : Correct   *Marks : 1/1*

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

Answer

8

Status : Correct   *Marks : 1/1*

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

Answer

Array

Status : Correct   *Marks : 1/1*

8. What is the primary disadvantage of linear probing?

Answer

Clustering

Status : Correct   **Marks : 1/1**

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

Answer

Folding shift method

Status : Wrong   **Marks : 0/1**

10. 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

1

Status : Correct   **Marks : 1/1**

11. Which situation causes clustering in linear probing?

Answer

Poor hash function

Status : Wrong   **Marks : 0/1**

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

Answer

A prime number

Status : Correct   **Marks : 1/1**

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

Answer

index = (index + 1) % size;

Status : Correct   *Marks : 1/1*

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

Answer

It divides the key into parts and adds them.

Status : Correct   *Marks : 1/1*

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

Answer

Index goes out of bounds

Status : Correct   *Marks : 1/1*

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

Answer

O(n)

Status : Correct   *Marks : 1/1*

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

Answer

(i + 1) % table_size

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

Answer

h(k) = k % m

Status : Correct   *Marks : 1/1*

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

Answer

Mid-Square

Status : Correct   *Marks : 1/1*

20.   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

(k * k) % 100

Status : Wrong   *Marks : 0/1*

Rajalakshmi Engineering College

Name: Divya Dharshini K
Email: 241501053@rajalakshmi.edu.in
Roll no: 241501053
Phone: 9566773627
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1; // Initialize all slots to -1 (empty)
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size; // Calculate initial index using hash function
    int originalIndex = index; // Store original index for loop control

    // Loop until we find an empty slot or return to the original index
    while (table[index] != -1) {
        index = (index + 1) % size; // Move to the next index
        if (index == originalIndex) {
            // If we looped back to the original index, the table is full
```

```c
        return -1; // Indicate that the table is full
    }
}
    return index; // Return the index of the empty slot
}

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]; // Insert the roll number at the found index
        }
    }
}

void printTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", table[i]); // Print each slot in the hash table
    }
    printf("\n"); // New line at the end
}

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: Divya Dharshini K
Email: 241501053@rajalakshmi.edu.in
Roll no: 241501053
Phone: 9566773627
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```c
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1; // Initialize all slots to -1 (empty)
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size; // Calculate initial index using hash function
    int originalIndex = index; // Store original index for loop control

    // Loop until we find an empty slot or return to the original index
```

```c
    while (table[index] != -1) {
        if (table[index] == num) {
            return index; // Return index if the number is found
        }
        index = (index + 1) % size; // Move to the next index
        if (index == originalIndex) {
            break; // If we looped back to the original index, stop searching
        }
    }
    return -1; // Indicate that the number was not found
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for (int i = 0; i < n; i++) {
        int index = arr[i] % size; // Calculate initial index using hash function
        while (table[index] != -1) {
            index = (index + 1) % size; // Move to the next index
        }
        table[index] = arr[i]; // Insert the roll number at the found index
    }
}

int searchInHashTable(int table[], int size, int num) {
    return linearProbe(table, size, num) != -1; // Return true if found
}

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);
```

```c
        else
            printf("Value %d: Not Found\n", x);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Divya Dharshini K
Email: 241501053@rajalakshmi.edu.in
Roll no: 241501053
Phone: 9566773627
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```c
#define MAX_CONTACTS 50
#define MAX_NAME_LENGTH 11
#define MAX_PHONE_LENGTH 11

typedef struct {
    char name[MAX_NAME_LENGTH];
    char phone[MAX_PHONE_LENGTH];
} Contact;

typedef struct {
    Contact contacts[MAX_CONTACTS];
    int count;
} ContactList;

void initializeContactList(ContactList *list) {
    list->count = 0; // Initialize the contact list count to 0
}

int addContact(ContactList *list, const char *name, const char *phone) {
    if (list->count >= MAX_CONTACTS) {
        return -1; // Contact list is full
    }
    strcpy(list->contacts[list->count].name, name);
    strcpy(list->contacts[list->count].phone, phone);
    list->count++;
    return 0; // Success
}

int removeContact(ContactList *list, const char *name) {
    for (int i = 0; i < list->count; i++) {
        if (strcmp(list->contacts[i].name, name) == 0) {
            // Shift contacts to remove the contact
            for (int j = i; j < list->count - 1; j++) {
                list->contacts[j] = list->contacts[j + 1];
            }
            list->count--; // Decrease the count
            return 1; // Contact removed
        }
    }
    return 0; // Contact not found
}
```

```c
void printContactList(ContactList *list) {
    for (int i = 0; i < list->count; i++) {
        printf("Key: %s; Value: %s\n", list->contacts[i].name, list->contacts[i].phone);
    }
}

int main() {
    ContactList contactList;
    initializeContactList(&contactList);

    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        char name[MAX_NAME_LENGTH], phone[MAX_PHONE_LENGTH];
        scanf("%s %s", name, phone);
        addContact(&contactList, name, phone);
    }

    char key[MAX_NAME_LENGTH];
    scanf("%s", key);

    if (removeContact(&contactList, key)) {
        printf("The given key is removed!\n");
        printContactList(&contactList);
    } else {
        printf("The given key is not found!\n");
        printContactList(&contactList);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Divya Dharshini K
Email: 241501053@rajalakshmi.edu.in
Roll no: 241501053
Phone: 9566773627
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

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

#define MAX_FRUITS 15
#define MAX_NAME_LENGTH 20

typedef struct {
    char name[MAX_NAME_LENGTH];
    int score;
} Fruit;

typedef struct {
    Fruit fruits[MAX_FRUITS];
    int count;
```

```c
} FruitContest;

void initializeFruitContest(FruitContest *contest) {
    contest->count = 0; // Initialize the count of fruits to 0
}

int addFruit(FruitContest *contest, const char *name, int score) {
    if (contest->count >= MAX_FRUITS) {
        return -1; // Contest is full
    }
    strcpy(contest->fruits[contest->count].name, name);
    contest->fruits[contest->count].score = score;
    contest->count++;
    return 0; // Success
}

int findFruit(FruitContest *contest, const char *name) {
    for (int i = 0; i < contest->count; i++) {
        if (strcmp(contest->fruits[i].name, name) == 0) {
            return i; // Return the index if found
        }
    }
    return -1; // Not found
}

int main() {
    FruitContest contest;
    initializeFruitContest(&contest);

    int N;
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        char name[MAX_NAME_LENGTH];
        int score;
        scanf("%s %d", name, &score);
        addFruit(&contest, name, score);
    }

    char searchName[MAX_NAME_LENGTH];
    scanf("%s", searchName);

    int index = findFruit(&contest, searchName);
```

```c
    if (index != -1) {
        printf("Key \"%s\" exists in the dictionary.\n", searchName);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", searchName);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Divya Dharshini K
Email: 241501053@rajalakshmi.edu.in
Roll no: 241501053
Phone: 9566773627
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are provided with a collection of numbers, each represented by an array of integers. However, there's a unique scenario: within this array, one element occurs an odd number of times, while all other elements occur an even number of times. Your objective is to identify and return the element that occurs an odd number of times in this arrangement.

Utilize mid-square hashing by squaring elements and extracting middle digits for hash codes. Implement a hash table for efficient integer occurrence tracking.

Note: Hash function: squared = key * key.

Example

Input:

7

2 2 3 3 4 4 5

Output:

5

Explanation

The hash function and the calculated hash indices for each element are as follows:

2 -> hash(2*2) % 100 = 4

3 -> hash(3*3) % 100 = 9

4 -> hash(4*4) % 100 = 16

5 -> hash(5*5) % 100 = 25

The hash table records the occurrence of each element's hash index:

Index 4: 2 occurrences

Index 9: 2 occurrences

Index 16: 2 occurrences

Index 25: 1 occurrence

Among the elements, the integer 5 occurs an odd number of times (1 occurrence) and satisfies the condition of the problem. Therefore, the program outputs 5.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer representing the element that occurs an odd

number of times.

If no such element exists, print -1.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
2 2 3 3 4 4 5
Output: 5

Answer

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

#define MAX_SIZE 100

#define TABLE_SIZE 100

// Hash function to calculate the hash index
unsigned int hash(int key, int tableSize) {
    int squared = key * key; // Square the key
    return (squared % tableSize); // Return the hash index
}

// Function to find the element that occurs an odd number of times
int getOddOccurrence(int arr[], int size) {
    int hashTable[TABLE_SIZE] = {0}; // Initialize hash table to zero

    // Count occurrences using the hash function
    for (int i = 0; i < size; i++) {
        int index = hash(arr[i], TABLE_SIZE);
        hashTable[index]++;
    }

    // Find the element with an odd occurrence
    for (int i = 0; i < size; i++) {
```

```c
        int index = hash(arr[i], TABLE_SIZE);
        if (hashTable[index] % 2 != 0) {
            return arr[i]; // Return the element with odd occurrences
        }
    }

    return -1; // Return -1 if no such element exists
}

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

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

    printf("%d\n", getOddOccurrence(arr, n));

    return 0;
}
```

Status : Correct   *Marks : 10/10*