Rajalakshmi Engineering College

Name: Swethaa V A
Email: 241501258@rajalakshmi.edu.in
Roll no: 241501258
Phone: 7200023170
Branch: REC
Department: I AI & ML FC
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

// Function to initialize the hash table with -1
void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1;
    }
}

// Function to insert roll numbers into the hash table using linear probing
void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for (int i = 0; i < n; i++) {
        int roll = arr[i];
        int index = roll % size;

        // Linear probing for empty slot
        int startIndex = index;
        while (table[index] != -1) {
            index = (index + 1) % size;
            if (index == startIndex) {
                // Table is full, though per constraints, should not happen
                break;
            }
        }
        table[index] = roll;
    }
}

// Function to print the hash table
void printTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", table[i]);
    }
    printf("\n");
}

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: Swethaa V A
Email: 241501258@rajalakshmi.edu.in
Roll no: 241501258
Phone: 7200023170
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

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

// Initialize the hash table with -1
void initializeTable(int table[], int size) {
   for (int i = 0; i < size; i++) {
      table[i] = -1;
   }
}

// Insert roll numbers into hash table using linear probing
void insertIntoHashTable(int table[], int size, int arr[], int n) {
   for (int i = 0; i < n; i++) {
      int roll = arr[i];
```

```
            int index = roll % size;
            int startIndex = index;

            // Find an empty slot via linear probing
            while (table[index] != -1) {
                index = (index + 1) % size;
                if (index == startIndex) {
                    // Table is full, though per constraints, should not happen
                    break;
                }
            }
            table[index] = roll;
        }
    }

// Search for a roll number in the hash table
int searchInHashTable(int table[], int size, int key) {
    int index = key % size;
    int startIndex = index;

    // Probe until found or come back to start
    while (table[index] != -1) {
        if (table[index] == key) {
            return 1; // Found
        }
        index = (index + 1) % size;
        if (index == startIndex) {
            break; // Came full circle, not found
        }
    }
    return 0; // Not 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);
        else
            printf("Value %d: Not Found\n", x);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Swethaa V A
Email: 241501258@rajalakshmi.edu.in
Roll no: 241501258
Phone: 7200023170
Branch: REC
Department: I AI & ML FC
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

```
unsigned long hash(const char *str) {
   unsigned long hash = 5381;
   int c;
   while ((c = *str++))
      hash = ((hash << 5) + hash) + c;
   return hash;
}

void insertKeyValuePair(Dictionary *dict, const char *key, const char *value) {
   if (dict->size == dict->capacity) {
      dict->capacity *= 2;
      dict->pairs = (KeyValuePair *)realloc(dict->pairs, dict->capacity *
sizeof(KeyValuePair));
   }
   strcpy(dict->pairs[dict->size].key, key);
   strcpy(dict->pairs[dict->size].value, value);
   dict->size++;
}

int doesKeyExist(Dictionary *dict, const char *key) {
   for (int i = 0; i < dict->size; i++) {
      if (strcmp(dict->pairs[i].key, key) == 0) {
         return 1;
      }
   }
   return 0;
}

void removeKeyValuePair(Dictionary *dict, const char *key) {
   int found = 0;
   for (int i = 0; i < dict->size; i++) {
      if (found) {
         dict->pairs[i - 1] = dict->pairs[i];
      }
      if (!found && strcmp(dict->pairs[i].key, key) == 0) {
         found = 1;
      }
   }
   if (found) {
      dict->size--;
   }
}

void printDictionary(Dictionary *dict) {
   for (int i = 0; i < dict->size; i++) {
      printf("Key: %s; Value: %s\n", dict->pairs[i].key, dict->pairs[i].value);
   }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Swethaa V A
Email: 241501258@rajalakshmi.edu.in
Roll no: 241501258
Phone: 7200023170
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

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: Swethaa V A
Email: 241501258@rajalakshmi.edu.in
Roll no: 241501258
Phone: 7200023170
Branch: REC
Department: I AI & ML FC
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

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

#define MAX_SIZE 100

int getOddOccurrence(int arr[], int n) {
   // Initialize hash array of size 100 with zeros
   int hashTable[100] = {0};
   int elementMap[1000]; // To store elements corresponding to each hash index
   // Initialize elementMap to a sentinel value
   for (int i = 0; i < 1000; i++) {
      elementMap[i] = -1;
   }

   for (int i = 0; i < n; i++) {
      int key = arr[i];
      // Compute hash: hash = middle digits of key*key
      long long squared = (long long)key * (long long)key;
      int hashIndex = (int)((squared / 10) % 100); // Extract middle digits
      // Increment occurrence count at hashIndex
      hashTable[hashIndex]++;
      // Store element at first occurrence
      if (elementMap[hashIndex] == -1) {
         elementMap[hashIndex] = key;
      }
```

```
    }

    // Find the element with odd occurrence
    for (int i = 0; i < 100; i++) {
        if (hashTable[i] % 2 != 0 && elementMap[i] != -1) {
            // Verify the actual element occurrence in array
            int count = 0;
            for (int j = 0; j < n; j++) {
                if (arr[j] == elementMap[i]) {
                    count++;
                }
            }
            if (count % 2 != 0) {
                return elementMap[i];
            }
        }
    }
    return -1; // No such element found
}

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***