Rajalakshmi Engineering College

Name: Nivedhitha K
Email: 240701371@rajalakshmi.edu.in
Roll no: 240701371
Phone: 9790413580
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

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
```

```
#include <stdio.h>

#define TABLE_SIZE 100

// Structure for key-count pair
typedef struct {
    int key;
    int count;
} HashItem;

// Each bucket can hold up to 10 items for simplicity
HashItem hashTable[TABLE_SIZE][10];
int bucketSizes[TABLE_SIZE] = {0};

// Mid-square hash function
unsigned int hash(int key) {
    int squared = key * key;
    return squared % 100; // Use last 2 digits as index
}
```

```
// Function to find the element with odd occurrence
int getOddOccurrence(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        int index = hash(arr[i]);
        int found = 0;

        // Check if element already exists in the bucket
        for (int j = 0; j < bucketSizes[index]; j++) {
            if (hashTable[index][j].key == arr[i]) {
                hashTable[index][j].count++;
                found = 1;
                break;
            }
        }

        // If not found, add new element
        if (!found) {
            int pos = bucketSizes[index];
            hashTable[index][pos].key = arr[i];
            hashTable[index][pos].count = 1;
            bucketSizes[index]++;
        }
    }

    // Look for element with odd occurrence
    for (int i = 0; i < TABLE_SIZE; i++) {
        for (int j = 0; j < bucketSizes[i]; j++) {
            if (hashTable[i][j].count % 2 != 0) {
                return hashTable[i][j].key;
            }
        }
    }

    return -1; // If no 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**