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

unsigned int hash(int key, int tableSize) {
    int squared = key * key;
    return squared % tableSize;
}

int getOddOccurrence(int arr[], int size) {
    int table[MAX_SIZE] = {0};
    int valueTable[MAX_SIZE] = {0};

    for (int i = 0; i < size; i++) {
        int key = arr[i];
        int index = hash(key, MAX_SIZE);

        if (valueTable[index] == 0 || valueTable[index] == key) {
            valueTable[index] = key;
            table[index]++;
        } else {
            int original_index = index;
            do {
                index = (index + 1) % MAX_SIZE;
```

```
            if (valueTable[index] == 0 || valueTable[index] == key) {
                valueTable[index] = key;
                table[index]++;
                break;
            }
        } while (index != original_index);
    }
}

for (int i = 0; i < size; i++) {
    int key = arr[i];
    int index = hash(key, MAX_SIZE);

    int original_index = index;
    do {
        if (valueTable[index] == key) {
            if (table[index] % 2 == 1) {
                return key;
            }
            break;
        }
        index = (index + 1) % MAX_SIZE;
    } while (index != original_index);
}

return -1;
}

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**