

# Rajalakshmi Engineering College

Name: Jeffery Antony J  
Email: 241901041@rajalakshmi.edu.in  
Roll no: 241901041  
Phone: 7305663808  
Branch: REC  
Department: I CSE (CS) FA  
Batch: 2028  
Degree: B.E - CSE (CS)

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 ->  $\text{hash}(2*2) \% 100 = 4$

3 ->  $\text{hash}(3*3) \% 100 = 9$

4 ->  $\text{hash}(4*4) \% 100 = 16$

5 ->  $\text{hash}(5*5) \% 100 = 25$

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
```

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

```
#define TABLE_SIZE 100
```

```
typedef struct {
    int key;
    int count;
} HashEntry;
```

```
int findIndex(HashEntry hashTable[], int key) {
    int square = key * key;
    int hashIndex = square % TABLE_SIZE;
```

```
    while (hashTable[hashIndex].count != 0 && hashTable[hashIndex].key != key) {
        hashIndex = (hashIndex + 1) % TABLE_SIZE;
```

```
    }
    return hashIndex;
```

```
}
```

```
int main() {
    int n, arr[20];
    HashEntry hashTable[TABLE_SIZE] = {0};
```

```
    scanf("%d", &n);
```

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

```
        int idx = findIndex(hashTable, arr[i]);
```

```
        if (hashTable[idx].count == 0) {
            hashTable[idx].key = arr[i];
```

```
        }
```

```
        hashTable[idx].count++;
```

```
    }
```

```
    for (int i = 0; i < TABLE_SIZE; i++) {
        if (hashTable[i].count % 2 == 1) {
            printf("%d\n", hashTable[i].key);
            return 0;
```

```
        }
    }
```

```
    printf("-1\n");
```

```
    return 0;
```

```
}
```

```
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**