

# Rajalakshmi Engineering College

Name: paranidharan R  
Email: 240801238@rajalakshmi.edu.in  
Roll no: 240801238  
Phone: 9360861582  
Branch: REC  
Department: I ECE AF  
Batch: 2028  
Degree: B.E - ECE

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

// You are using GCC
void initializeTable(int table[], int size) {
    for(int i=0;i<size;i++){
        table[i]=-1;
    }
}

int linearProbe(int table[], int size, int num) {
```

```

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for(int i=0;i<n;i++){
        int index=arr[i]%size;
        int j=index;

        while(table[j]!=-1){
            j=(j+1)%size;
            if(j==index){
                return;
            }
            //table[j]=arr[i];
        }
        table[j] = arr[i];
    }
}

```

```

int searchInHashTable(int table[], int size, int num) {

```

```

    int index = num % size;

```

```

    do {
        if(table[index] == -1) {
            return 0;
        }
        if(table[index] == num) {
            return 1;
        }
        index = (index + 1) % size;
    } while(index != (num % size));

```

```

}

```

```

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**