

**NAME: AMBRISH KUMAR A**

**REG NO: 241801017**

## **DIVIDE AND CONQUER**

### **PROGRAM 1:**

**Question 1** | Correct | Mark 1.00 out of 1.00 | [Flag question](#)

#### **Problem Statement**

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.

Input Format

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

Output Format

First Line Contains Integer – Number of zeroes present in the given array.

**Answer:** (penalty regime: 0 %)

```
1  #include <stdio.h>
2
3  int find_first_zero(int arr[], int low, int high) {
4      int result = -1;
5      while (low <= high) {
6          int mid = (low + high) / 2;
7          if (arr[mid] == 0) {
8              result = mid;
9              high = mid - 1;
10         } else {
11             low = mid + 1;
12         }
13     }
14     return result;
15 }
16
17 int main() {
18     int m;
19     scanf("%d", &m);
20
21     int arr[m];
22     for (int i = 0; i < m; i++) {
23         scanf("%d", &arr[i]);
24     }
25
26     int first_zero_index = find_first_zero(arr, 0, m - 1);
27
28     if (first_zero_index == -1) {
29         printf("0\n");
30     } else {
31         printf("%d", m - first_zero_index);
32     }
33
34     return 0;
35 }
36
```

	Input	Expected	Got	
✓	5	2	2	✓
	1			
	1			
	1			
	0			
	0			
✓	10	0	0	✓
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			

✓	8	8	8	✓
	0			
	0			
	0			
	0			
	0			
	0			
	0			
	0			
	0			
✓	17	2	2	✓
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	0			
	0			

# PROGRAM 2:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than  $\lfloor n / 2 \rfloor$  times. You may assume that the majority element always exists in the array.

## Example 1:

Input: `nums = [3,2,3]`

Output: 3

## Example 2:

Input: `nums = [2,2,1,1,1,2,2]`

Output: 2

## Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 104`
- `-231 <= nums[i] <= 231 - 1`

## For example:

Input	Result
3 3 2 3	3
7 2 2 1 1 1 2 2	2

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 v int majorityElement(int nums[], int numsSize) {
4     int count = 0, candidate = 0;
5
6 v     for (int i = 0; i < numsSize; i++) {
7 v         if (count == 0) {
8             candidate = nums[i];
9             count = 1;
10 v        } else if (nums[i] == candidate) {
11            count++;
12 v        } else {
13            count--;
14        }
15    }
16    return candidate;
17 }
18
19 v int main() {
20     int n;
21     scanf("%d", &n);
22
23     int nums[n];
24 v     for (int i = 0; i < n; i++) {
25         scanf("%d", &nums[i]);
26     }
27
28     int result = majorityElement(nums, n);
29     printf("%d\n", result);
30
31     return 0;
32 }
33
```

	Input	Expected	Got	
✓	3	3	3	✓
	3 2 3			

Passed all tests! ✓

# PROGRAM 3:

Question 1 | Correct Mark 1.00 out of 1.00 Flag question

## Problem Statement:

Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

## Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Value for x

## Output Format

First Line Contains Integer – Floor value for x

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 int findFloor(int arr[], int n, int x) {
4     int low = 0, high = n - 1;
5     int floor = -1;
6
7     while (low <= high) {
8         int mid = low + (high - low) / 2;
9
10        if (arr[mid] == x) {
11            return arr[mid];
12        }
13        else if (arr[mid] < x) {
14            floor = arr[mid];
15            low = mid + 1;
16        }
17        else {
18            high = mid - 1;
19        }
20    }
21
22    return floor;
23 }
24
25 int main() {
26     int n, x;
27     scanf("%d", &n);
28     int arr[n];
29
30     for (int i = 0; i < n; i++) {
31         scanf("%d", &arr[i]);
32     }
33     scanf("%d", &x);
34
35     int result = findFloor(arr, n, x);
36     printf("%d\n", result);
37
38     return 0;
39 }
40
```

	Input	Expected	Got	
✓	6	2	2	✓
	1			
	2			
	8			
	10			
	12			
	19			
	5			
✓	5	85	85	✓
	10			
	22			
	85			
	108			
	129			
	100			
✓	7	9	9	✓
	3			
	5			
	7			
	9			
	11			
	13			
	15			
	10			

Passed all tests! ✓

# PROGRAM 4:

Question 1 | Correct | Mark 1.00 out of 1.00 | [Flag question](#)

## Problem Statement:

Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".

Note: Write a Divide and Conquer Solution

## Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Sum Value

## Output Format

First Line Contains Integer – Element1

Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 int elem1 = 0, elem2 = 0;
4
5 int findPair(int arr[], int left, int right, int x) {
6     if (left >= right) {
7         return 0;
8     }
9
10    int sum = arr[left] + arr[right];
11    if (sum == x) {
12        elem1 = arr[left];
13        elem2 = arr[right];
14        return 1;
15    }
16    else if (sum < x) {
17        return findPair(arr, left + 1, right, x);
18    }
19    else {
20        return findPair(arr, left, right - 1, x);
21    }
22 }
23
24 int main() {
25     int n, x;
26     scanf("%d", &n);
27     int arr[n];
28
29     for (int i = 0; i < n; i++) {
30         scanf("%d", &arr[i]);
31     }
32     scanf("%d", &x);
33
34     if (findPair(arr, 0, n - 1, x)) {
35         printf("%d\n%d\n", elem1, elem2);
36     } else {
37         printf("No\n");
38     }
39
40     return 0;
41 }
```

	Input	Expected	Got	
✓	4	4	4	✓
	2	10	10	
	4			
	8			
	10			
	14			
✓	5	No	No	✓
	2			
	4			
	6			
	8			
	10			
	100			

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.



## PROGRAM 5:

**Question 1** | Correct Mark 1.00 out of 1.00 [Flag question](#)

Write a Program to Implement the Quick Sort Algorithm

Input Format:

The first line contains the no of elements in the list-n

The next n lines contain the elements.

Output:

Sorted list of elements

**For example:**

Input	Result
5	12 34 67 78 98
67 34 12 98 78	

Answer:

```
1 #include <stdio.h>
2
3 void swap(int *a, int *b) {
4     int temp = *a;
5     *a = *b;
6     *b = temp;
7 }
8
9 int partition(int arr[], int low, int high) {
10     int pivot = arr[high];
11     int i = low - 1;
12
13     for (int j = low; j <= high - 1; j++) {
14         if (arr[j] <= pivot) {
15             i++;
16             swap(&arr[i], &arr[j]);
17         }
18     }
19     swap(&arr[i + 1], &arr[high]);
20     return i + 1;
21 }
22
23 void quickSort(int arr[], int low, int high) {
24     if (low < high) {
25         int pi = partition(arr, low, high);
26
27         quickSort(arr, low, pi - 1);
28         quickSort(arr, pi + 1, high);
29     }
30 }
31
32 int main() {
33     int n;
34     scanf("%d", &n);
35
36     int arr[n];
37     for (int i = 0; i < n; i++) {
38         scanf("%d", &arr[i]);
39     }
40
41     quickSort(arr, 0, n - 1);
42
43     for (int i = 0; i < n; i++) {
44         printf("%d ", arr[i]);
45     }
46     printf("\n");
47
48     return 0;
49 }
50
```

	Input	Expected	Got	
✓	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✓
✓	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	✓
✓	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.