Question 1 Correct Mark 1.00 out of 1.00

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 %)

```c
#include <stdio.h>
int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    int low = 0, high = m - 1;
    int first_zero_index = m;

    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == 0) {
            first_zero_index = mid;
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    printf("%d\n", m - first_zero_index);
    return 0;
}
```

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

Question 1 Correct Mark 1.00 out of 1.00

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

The majority element is the element that appears more than ⌊n / 2⌋ 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 * 10^4
- -2^{31} <= nums[i] <= 2^{31} - 1

For example:

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

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);

    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    int candidate = nums[0];
    int count = 1;

    for (int i = 1; i < n; i++) {
        if (nums[i] == candidate) {
            count++;
        } else {
            count--;
            if (count == 0) {
                candidate = nums[i];
                count = 1;
            }
        }
    }
    printf("%d\n", candidate);

    return 0;
}

```

Check

Question 1

Correct Mark 1.00 out of 1.00

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 %)

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int x;
    scanf("%d", &x);
    int low = 0, high = n - 1;
    int floor_val = -1;
    while(low <= high) {
        int mid = (low + high) / 2;
        if(arr[mid] == x) {
            floor_val = arr[mid];
            break;
        }
        else if(arr[mid] < x) {
            floor_val = arr[mid];
            low = mid + 1;
        }
        else {
            high = mid - 1;
        }
    }
    printf("%d\n", floor_val);
    return 0;
}

```

	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓

Question 1

Correct Mark 1.00 out of 1.00

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 %)

```
#include <stdio.h>
void findPair(int arr[], int low, int high, int x) {
    if (low >= high) {
        printf("No\n");
        return;
    }
    int sum = arr[low] + arr[high];
    if (sum == x) {
        printf("%d\n%d\n", arr[low], arr[high]);
        return;
    }
    else if (sum < x) {
        findPair(arr, low + 1, high, x);
    }
    else {
        findPair(arr, low, high - 1, x);
    }
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int x;
    scanf("%d", &x);
    findPair(arr, 0, n - 1, x);
    return 0;
}

```

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

Question 1 Not complete Mark 1.00 out of 1.00

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 67 34 12 98 78	12 34 67 78 98

Answer:

```
#include <stdio.h>
void quickSort(int arr[], int low, int high) {
    if (low >= high) return;
    int pivot = arr[high];
    int i = low;
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
        }
    }
    int temp = arr[i];
    arr[i] = arr[high];
    arr[high] = temp;
    quickSort(arr, low, i - 1);
    quickSort(arr, i + 1, high);
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for(int i=0; i<n; i++)
        scanf("%d", &arr[i]);
    quickSort(arr, 0, n-1);
    for(int i=0; i<n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    return 0;
}

```

Check

	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	✓