DIVIDE AND CONQUER:-

1) **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 count(int a[], int l, int h){
    if(l>h){
        return 0;
    }

    if(l==h){
        if(a[l]==0){
            return 1;
        }
        else{
            return 0;
        }
    }

    int mid = (l+h)/2;

    int lm = count(a, l, mid);
    int rm = count(a, mid+1, h);
    return rm+lm;

}

int main(){
    int m;
    scanf("%d",&m);
    int a[m];
    for(int i=0;i<m;i++){
        scanf("%d",&a[i]);
    }

    int ans = count(a, 0, m-1);
    printf("%d",ans);

}
```

	Input	Expected	Got	
✔	5 1 1 1 0 0	2	2	✔
✔	10 1 1 1 1 1 1 1 1 1 1	0	0	✔
✔	8 0 0 0 0 0 0 0 0	8	8	✔

2) 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 %)

```c
#include<stdio.h>
int divide(int a[],int l,int r,int s)
{
    if(l==r)
    {
        return a[l];
    }
    int mid=(l+r)/2;
    int left=divide(a,l,mid,s);
    int right=divide(a,mid+1,r,s);
    int lcount=0,rcount=0;
    for(int i=0;i<s;i++)
    {
        if(a[i]==left)
            lcount++;
        if(a[i]==right)
            rcount++;
    }
    if(lcount>(s/2))
        return left;
    else
        return right;
}
int main()
{
    int size;
    scanf("%d",&size);
    int arr[size];
    for(int i=0;i<size;i++)
    {
        scanf("%d",&arr[i]);
    }
    int low=0,high=size-1;
    int majority=divide(arr,low,high,size);
    printf("%d",majority);
}
```

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3) **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 %)

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

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

Passed all tests! ✔

4) **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 %)

```c
#include<stdio.h>
void sum(int a[],int l,int r,int x){
    if (l >= r) {
        printf("No\n");
        return;
    }
    int asum=a[l]+a[r];
    if(asum==x){
        printf("%d\n",a[l]);
        printf("%d\n",a[r]);
    }else if(asum<x){
        sum(a,l+1,r,x);
    }else{
        sum(a,l,r-1,x);
    }
}

int main(){
    int n,x;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    scanf("%d",&x);
    sum(a,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	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

5) 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

```c
#include <stdio.h>
void QuickSort(int a[], int left, int right) {
    int i, j, temp, pivot;
    if (left < right) {
        pivot = left;
        i = left + 1;
        j = right;
        while (i <= j) {
            while (i <= right && a[i] < a[pivot]) {
                i++;
            }
            while (a[j] > a[pivot]) {
                j--;
            }
            if (i <= j) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
                i++;
                j--;
            }
        }
        temp = a[pivot];
        a[pivot] = a[j];
        a[j] = temp;
        QuickSort(a, left, j - 1);
        QuickSort(a, j + 1, right);
    }
}
```

```c
int main() {
    int i, n;
    scanf("%d", &n);
    int a[n];
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    QuickSort(a, 0, n - 1);
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    return 0;
}
```

	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! ✔