NAME: SREYA G **REGISTER NO.:**230701334

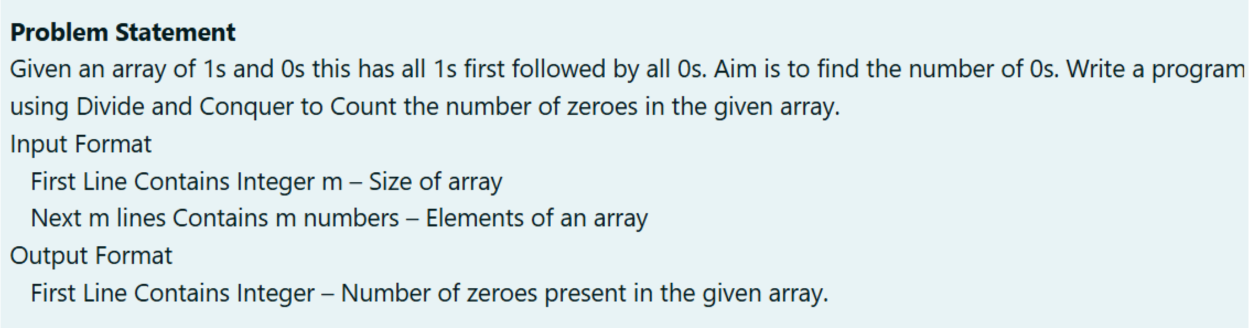
CLASS: CSE F **DATE:** 29/08/2024

EX - 4:

DIVIDE AND CONQUER:

PROBLEM 1:

AIM:



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.

ALGORITHM:

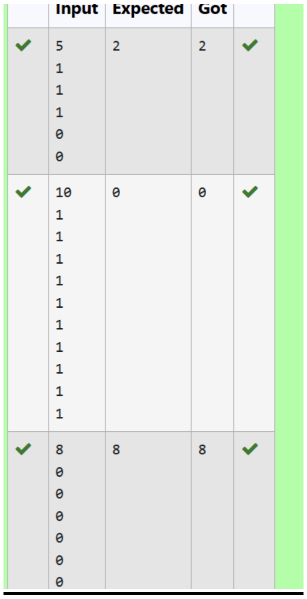
```
1. Input the array size (n) and the array elements.
2. Initialize a counter (c) to 0 for storing the count of zeros.
3. Define a recursive function (find):
   - If the leftmost element of the current segment is 0:
     - Increment c by the number of elements in the segment.
   - Otherwise, divide the segment into two halves and recurse.
4. Call find on the full array(left = 0, right = n - 1).
5. Output the counter (c).
```

CODE:

```
#include<stdio.h>
int c = 0;
```

```
void find(int a[],int left,int right)
{
    if(a[left] == 0)
    {
        c += (right-left+1);
    }
    else
    {
        if(left < right)
        {
            int m = (left + right)/2;
            find(a,left,m);
            find(a,m+1,right);
        }
    }
}
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i = 0;i < n;i++)
    {
        scanf("%d",&a[i]);
    }
    find(a,0,n - 1);
    printf("%d",c);
}
```

OUTPUT:



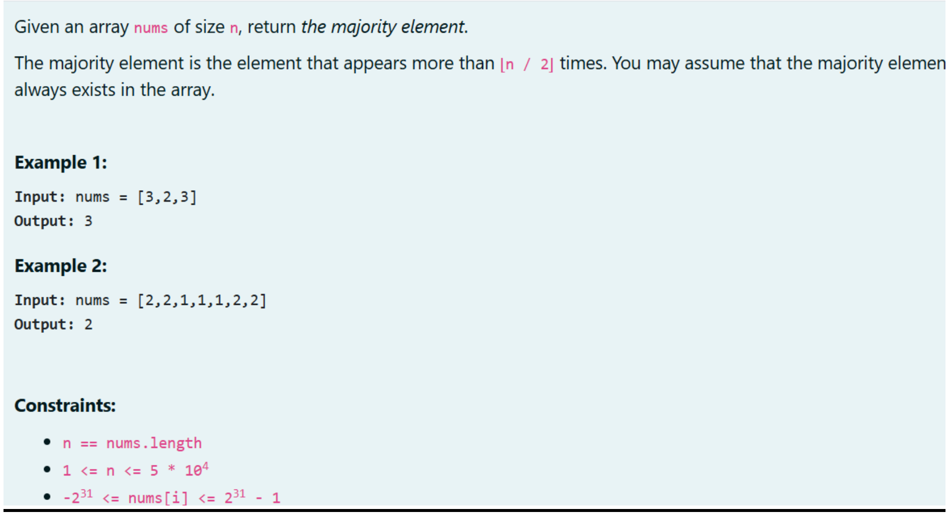
	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1 1 1	0	0	✓
✓	8 0 0 0 0 0 0	8	8	✓

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 2:

AIM:



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 elemen always exists in the array.

Example 1:

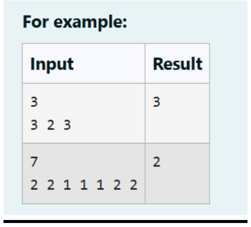
```
Input: nums = [3,2,3]
Output: 3
```

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

ALGORITHM:

1. Input nums (size) and array arr.

2. Initialize counter c = 0.

3. Recursive Function C(a, l, r, k):

   - Count occurrences of k in a[l:r].

4. Set b = arr[0] as the candidate element.

5. If C(arr, 0, nums, b) > nums/2, print b.

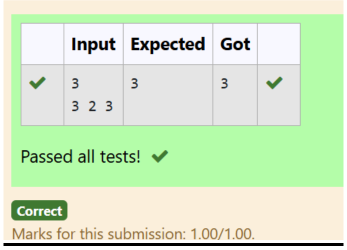
6. Else, check first half for a different element and print b.

CODE:

```
#include<stdio.h>

int c = 0;

int C(int a[],int l,int r,int k)
```

```
{
    int m = l+(r-l)/2;
    if(a[m] == k)
    {
        c++;
    }
    else
    {
        C(a,l,m,k);
        C(a,m+1,r,k);
    }
      return c;
}


int main()
{
    int nums;
    scanf("%d",&nums);
    int arr[nums];
    for(int i = 0;i < nums;i++)
    {
        scanf("%d",&arr[i]);
    }
    int b = arr[0];
    if(C(arr,0,nums,b) > nums/2)
    {
     printf("%d",b);
  }
  else
  {
    for(int i = 0;i < nums/2;i++)
     {
      if(arr[i] != b)
      {
        printf("%d",b);
        break;

      }
    }
  }
}
```

OUTPUT:



	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

Correct

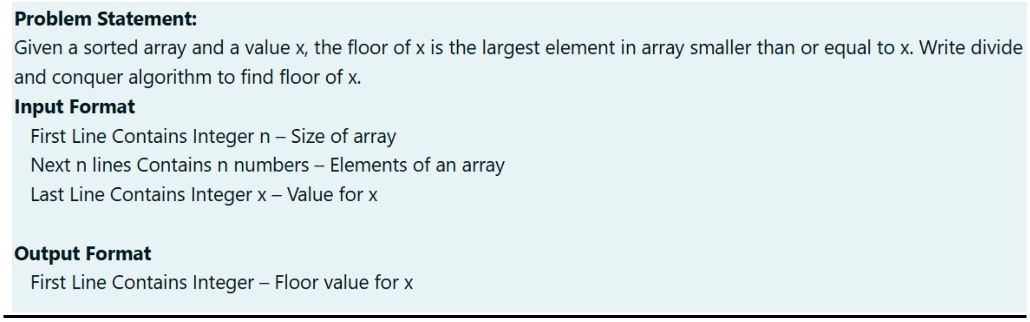
Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 3:

AIM:



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

ALGORITHM:

1. Input n (size) and array a[].

2. Input divisor.

3. Define recursive function maxfloor(a, left, right, divisor):

   - Calculate the middle index mid.

   - If a[mid] divided by divisor is 0 and greater than highest, update highest.

   - Recursively check the left and right segments.

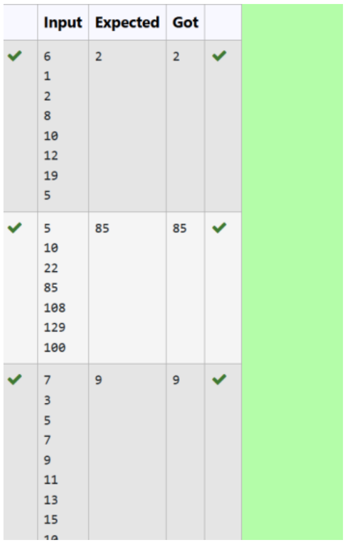
4. Call maxfloor(a, 0, n, divisor)

5. Output the value of highest.

CODE:

```
#include<stdio.h>

int highest = 0;

int maxfloor(int a[], int left, int right,int divisor)

{

    if(left < right)

    {

        int mid = (left+right-1)/2;

        if(a[mid]/divisor == 0 && highest < a[mid])

        {

            highest = a[mid];

        }
        maxfloor(a,left,mid,divisor);
        maxfloor(a,mid+1,right,divisor);
    }
    return highest;
}

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

    int divisor;
    scanf("%d",&divisor);
    printf("%d",maxfloor(a,0,n,divisor));
}
```

OUTPUT:



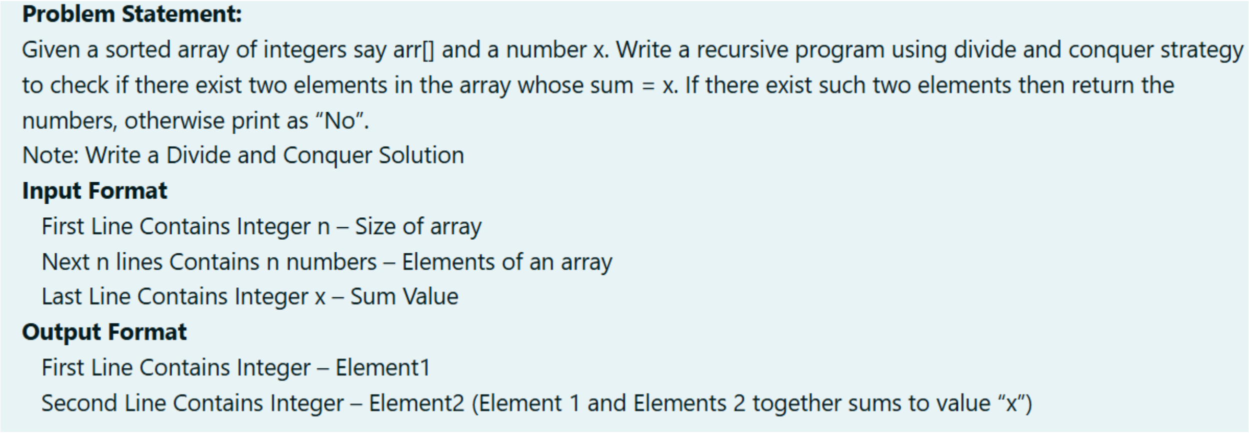
	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 108 129 100	85	85	✓
✓	7 3 5 7 9 11 13 15 10	9	9	✓

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 4:

AIM:



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")

ALGORITHM:

1. Input size n, array arr[], and sum x.

2. Define recursive function sum(arr, l, r, s):

- Check if the sum of arr[mid] and arr[r] equals s. If true, set p = arr[mid] and q = arr[r].

- Recursively check the array from l to r-1.

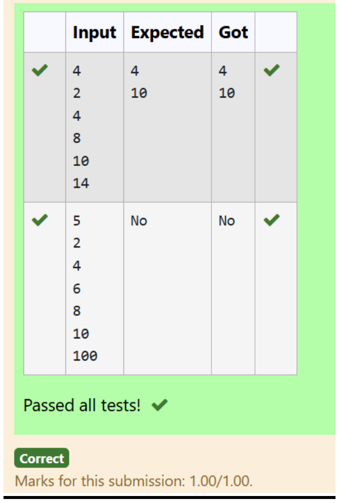
3. Call sum(arr, 0, n-1, x).

4. If no pair found, output "No". Otherwise, output p and q.

CODE:

```
#include<stdio.h>
int p = 0,q = 0;
int sum(int arr[],int l,int r,int s)
{
    if(l<r)
    {
        int mid = (l+r)/2;
        if(arr[mid]+arr[r] == s)
        {
            p =arr[mid];
            q = arr[r];
            return 1;
        }
        sum(arr,l,r-1,s);
    }
    return 0;
}
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 y = sum(arr,0,n-1,x);
    if(y == 0)
    {
        printf("%s","No");
    }

    else
    {
        printf("%d\n%d",p,q);
    }
}
```

OUTPUT:



	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

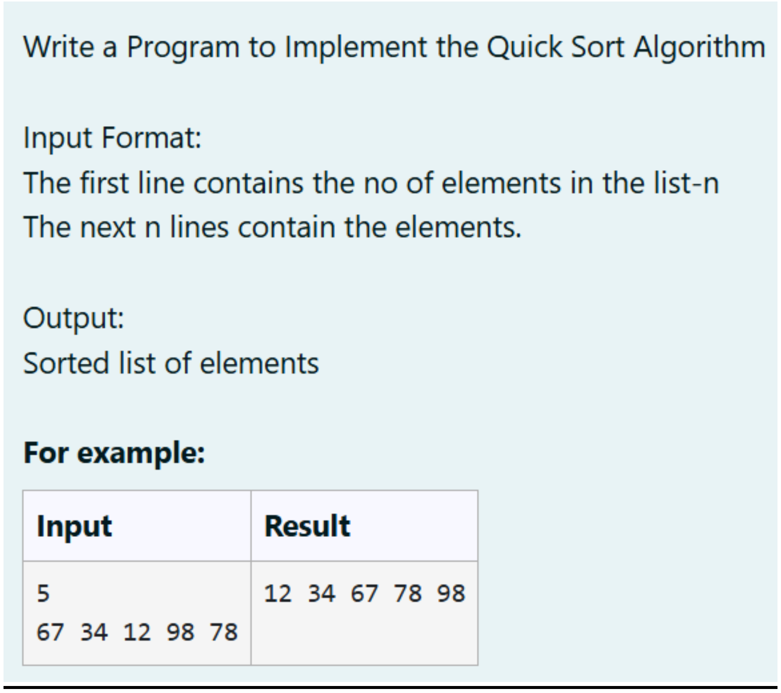
Marks for this submission: 1.00/1.00.

RESULT:

Thus the code is executed successfully and gives the expected output.

PROBLEM 5:

AIM:



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

ALGORITHM:

1. Input array arr[] and size n.

2. Define quickSort(arr, left, right):

   - Choose a pivot element (arr[mid]).

   - Partition the array: elements less than the pivot on the left, elements greater on the right.

   - Recursively sort the left and right subarrays.

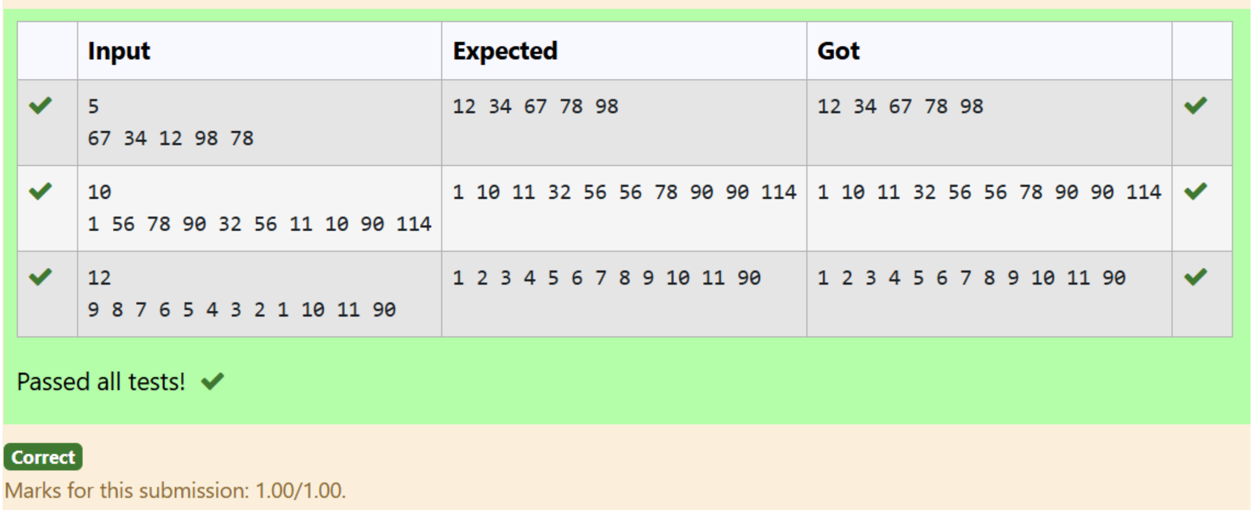
3. Call quickSort(arr, 0, n-1).

4. Output the sorted array.

CODE:

```
#include<stdio.h>
void quickSort(int arr[],int left,int right)
{
    if(left < right)
    {
        int pivot = (left + right)/2;
        int i = left;
        int j = right;
        while(i < j)
        {
            while(arr[pivot] >= arr[i])
            {
              i++;
            }

            while(arr[pivot] < arr[j])
            {
                j--;
            }
            if(i <= j)
            {
                int t = arr[i];
                arr[i] = arr[j];
                arr[j] = t;

            }
        }
    int t = arr[j];
    arr[j] = arr[pivot];
    arr[pivot] = t;
    quickSort(arr,left+1,right);
    }
}

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]);
    }
}
```

OUTPUT:



	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.

RESLUT:

Thus the code is executed successfully and gives the expected output.