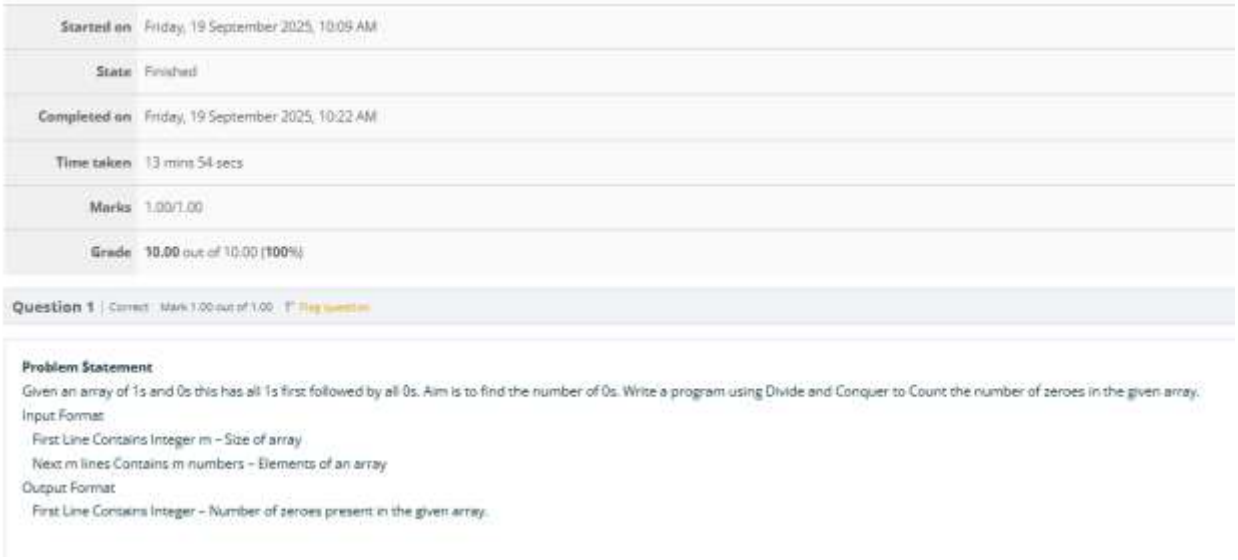
Divide and conquer

Roll no.: 241801152

Name : Manoj PT

1-Number of Zeros in a Given Array



Started on	Friday, 19 September 2025, 10:09 AM
State	Finished
Completed on	Friday, 19 September 2025, 10:22 AM
Time taken	13 mins 54 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100%**)

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.

```c
#include<stdio.h>

int findzero(int a[],int low,int high){
    if(low>high)
        return -1;
    int mid=(low+high)/2;
    if(a[mid]==0 &&(mid==0 || a[mid-1]==1))
        return mid;
    if(a[mid]==1)
        return findzero(a,mid+1,high);
    return findzero(a,low,mid-1);
}
int count(int a[],int m){
    int total=findzero(a,0,m-1);
    if(total==-1){
        return 0;
    }

    return m-total;
}

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

	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	✔
✔	17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2-Majority Element

Started on	Friday, 19 September 2025, 10:24 AM
State	Finished
Completed on	Friday, 19 September 2025, 10:41 AM
Time taken	16 mins 41 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

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 `⌊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

```c
#include<stdio.h>
int count(int a[],int m){
    int num=-1,count=0;
    for(int i=0;i<m;i++){
        if(count==0){
            num=a[i];
            count=1;

        }


        else if(a[i]==num){
            count++;
        }
        else{
            count--;
        }
    }
    return num;

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

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

3-Finding Floor Value

Started on	Friday, 19 September 2025, 10:42 AM
State	Finished
Completed on	Friday, 19 September 2025, 10:57 AM
Time taken	15 mins 24 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

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

```c
#include<stdio.h>
int find(int a[],int low,int high,int x){
    if(low>high){
        return -1;
    }
    int mid=(low+high)/2;
    if(a[mid]==x){
        return a[mid];
    }
    else if(a[mid]>x){
        return find(a,low,mid-1,x);
    }
    else{
        int total=find(a,mid+1,high,x);
        if(total==-1 || (total>x)){
            return a[mid];
        }
        else{
            return total;
        }
    }

}
int main(){
    int m;
    scanf("%d ",&m);
    int a[m];
    for(int i=0;i<m;i++){
        scanf("%d ",&a[i]);
    }
    int x;
    scanf("%d ",&x);
    int final=find(a,0,m-1,x);
    printf("%d ",final);
}
```

	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! ✔

Correct

Marks for this submission: 1.00/1.00.

4-Two Elements sum to x

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 find(int a[],int low,int high,int x){
    if(low>=high){
        printf("No");
        return;
    }
    int sum=a[low]+a[high];
    if(sum==x){
        printf("%d\n%d\n",a[low],a[high]);
    }
    else if(sum<x){
        return find(a,low+1,high,x);
    }
    else{
        return find(a,low,high+1,x);
    }

}
int main(){
    int n;
    scanf("%d ",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d ",&a[i]);
    }
    int x;
    scanf("%d ",&x);
    find(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-Implementation of Quick Sort

Started on	Friday, 19 September 2025, 6:19 PM
State	Finished
Completed on	Friday, 19 September 2025, 6:27 PM
Time taken	8 mins 33 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

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 67 34 12 98 78	12 34 67 78 98

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int p = partition(arr, low, high);
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 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]);
    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! ✔