

230701016

M. AISHWARYA

CSE-A

1.

AIM-

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.

CODE-

```
1 #include <stdio.h>
2 int findFirstZero(int arr[], int low, int high) {
3     if (low > high)
4         return -1;
5     int mid = low + (high - low) / 2;
6     if (arr[mid] == 0 && (mid == 0 || arr[mid - 1] == 1))
7         return mid;
8     return arr[mid] == 1 ? findFirstZero(arr, mid + 1, high) : findFirstZero(arr, low, mid - 1);
9 }
10
11 int countZeroes(int arr[], int size) {
12     int firstZero = findFirstZero(arr, 0, size - 1);
13     return firstZero == -1 ? 0 : size - firstZero;
14 }
15
16 int main() {
17     int m;
18     scanf("%d", &m);
19     int arr[m];
20     for (int i = 0; i < m; i++)
21     {
22         scanf("%d", &arr[i]);
23     }
24     printf("%d\n", countZeroes(arr, m));
25     return 0;
26 }
27
```

INPUT-

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

OUTPUT-

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

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

Passed all tests! ✓

2.

AIM-

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.

CODE-

```
1 #include <stdio.h>
2 int find(int nums[], int n) {
3     int count = 0;
4     int t = 0;
5
6     for (int i = 0; i < n; i++)
7     {
8         if (count == 0) {
9             t = nums[i];
10        }
11        count += (nums[i] == t) ? 1 : -1;
12    }
13    return t;
14 }
15
16 int main() {
17     int n;
18     scanf("%d", &n);
19     int nums[n];
20     for (int i = 0; i < n; i++) {
21         scanf("%d", &nums[i]);
22     }
23     int majele = find(nums, n);
24     printf("%d\n", majele);
25     return 0;
26 }
27
```

INPUT-

nums = [3,2,3]

OUTPUT-

3

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

3.

AIM-

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.

CODE-

```
1 #include <stdio.h>
2 int findFloor(int arr[], int low, int high, int x) {
3     if (x < arr[low])
4         return -1;
5     if (x >= arr[high]) return arr[high];
6     int mid = low + (high - low) / 2;
7     if (arr[mid] == x) {
8         return arr[mid];
9     }
10    if (arr[mid] < x) {
11        if (mid + 1 <= high && arr[mid + 1] > x) {
12            return arr[mid];
13        }
14        return findFloor(arr, mid + 1, high, x);
15    }
16    return findFloor(arr, low, mid - 1, x);
17 }
18 int main() {
19     int n, x;
20     scanf("%d", &n);
21     int arr[n];
22     for (int i = 0; i < n; i++) {
23         scanf("%d", &arr[i]);
24     }
25     scanf("%d", &x);
26     int floor = findFloor(arr, 0, n - 1, x);
27     if (floor == -1) {
28         printf("No floor found\n");
29     } else {
30         printf("%d\n", floor);
31     }
32     return 0;
33 }
34 }
35 }
```

INPUT-

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-

First Line Contains Integer – Floor value for x

	Input	Expected	Got	
✓	6 1 2 3 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.

AIM-

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

CODE-

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

INPUT-

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-

First Line Contains Integer – Element1

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

	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.

AIM-

Write a Program to Implement the Quick Sort Algorithm.

CODE-

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

INPUT-

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

The next n lines contain the elements.

OUTPUT-

Sorted list of elements

	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.