

Divide and Conquer

1-Number of Zeros in a Given Array

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.

Answer: (penalty regime: 0 %)

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

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

2-Majority Element

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 $\lfloor n / 2 \rfloor$ 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 * 104`
- `-231 <= nums[i] <= 231 - 1`

For example:

| Input | Result |
|---------------|--------|
| 3 | 3 |
| 3 2 3 | |
| 7 | 2 |
| 2 2 1 1 1 2 2 | |

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 int majorityElement(int nums[], int n) {
4     int count = 0;
5     int candidate = 0;
6
7     for (int i = 0; i < n; i++) {
8         if (count == 0) {
9             candidate = nums[i];
10        }
11        if (nums[i] == candidate)
12            count++;
13        else
14            count--;
15    }
16    return candidate;
17 }
18
19 int main() {
20     int n;
21     scanf("%d", &n);
22
23     int nums[n];
24     for (int i = 0; i < n; i++)
25         scanf("%d", &nums[i]);
26
27     int result = majorityElement(nums, n);
28     printf("%d\n", result);
29
30     return 0;
31 }
```

| | Input | Expected | Got | |
|---|------------|----------|-----|---|
| ✓ | 3 3 2 3 | 3 | 3 | ✓ |
| | | | | |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

3-Finding Floor Value

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 the floor.

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 %)

```
1 #include <stdio.h>
2 v int findFloor(int arr[], int low, int high, int x) {
3     if (x < arr[low])
4         return -1;
5
6     if (x >= arr[high])
7         return arr[high];
8
9     int mid = (low + high) / 2;
10
11    if (arr[mid] == x)
12        return arr[mid];
13    if (mid < high && arr[mid] <= x && x < arr[mid + 1])
14        return arr[mid];
15    if (x < arr[mid])
16        return findFloor(arr, low, mid - 1, x);
17    return findFloor(arr, mid + 1, high, x);
18 }
19
20 v int main() {
21     int n;
22     scanf("%d", &n);
23
24     int arr[n];
25     for (int i = 0; i < n; i++)
26         scanf("%d", &arr[i]);
27
28     int x;
29     scanf("%d", &x);
30
31     int result = findFloor(arr, 0, n - 1, x);
32
33     if (result == -1)
34         printf("Floor does not exist\n");
35     else
36         printf("%d\n", result);
37
38     return 0;
39 }
40 }
```

| | 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 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 %)

```
1 #include <stdio.h>
2
3 v int findPair(int arr[], int left, int right, int x, int *a, int *b) {
4     if (left >= right)
5         return 0;
6
7     int sum = arr[left] + arr[right];
8
9 v     if (sum == x) {
10         *a = arr[left];
11         *b = arr[right];
12         return 1;
13     }
14
15     if (sum > x)
16         return findPair(arr, left, right - 1, x, a, b);
17     else
18         return findPair(arr, left + 1, right, x, a, b);
19 }
20
21 v int main() {
22     int n, x;
23     scanf("%d", &n);
24
25     int arr[n];
26     for (int i = 0; i < n; i++)
27         scanf("%d", &arr[i]);
28
29     scanf("%d", &x);
30
31     int a, b;
32     if (findPair(arr, 0, n - 1, x, &a, &b)) {
33         printf("%d\n%d\n", a, b);
34     } else {
35         printf("No\n");
36     }
37
38     return 0;
}
```

| | Input | Expected | Got | |
|---|-------|----------|-----|---|
| ✓ | 4 | 4 | 4 | ✓ |
| | 2 | 10 | 10 | |
| | 4 | | | |
| | 8 | | | |
| | 10 | | | |
| | 14 | | | |

| | | | | |
|---|-----|----|----|---|
| ✓ | 5 | No | No | ✓ |
| | 2 | | | |
| | 4 | | | |
| | 6 | | | |
| | 8 | | | |
| | 10 | | | |
| | 100 | | | |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

5-Implementation of Quick Sort

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 |

Answer:

```
1 #include <stdio.h>
2 void swap(int *a, int *b) {
3     int temp = *a;
4     *a = *b;
5     *b = temp;
6 }
7
8 int partition(int arr[], int low, int high) {
9     int pivot = arr[high];
10    int i = (low - 1);
11
12    for (int j = low; j < high; j++) {
13        if (arr[j] < pivot) {
14            i++;
15            swap(&arr[i], &arr[j]);
16        }
17    }
18
19    swap(&arr[i + 1], &arr[high]);
20    return (i + 1);
21 }
22
23 void quickSort(int arr[], int low, int high) {
24     if (low < high) {
25         int pi = partition(arr, low, high);
26         quickSort(arr, low, pi - 1);
27         quickSort(arr, pi + 1, high);
28     }
29 }
30
31 int main() {
32     int n;
33     scanf("%d", &n);
34
35     int arr[n];
36     for (int i = 0; i < n; i++) {
37         scanf("%d", &arr[i]);
38     }
39
40     quickSort(arr, 0, n - 1);
41
42     for (int i = 0; i < n; i++) {
43         printf("%d ", arr[i]);
44     }
45
46     return 0;
47 }
```

| | 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.