

DIVIDE AND CONQUER

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
3  int countZeroes(int arr[], int m) {
4      int left = 0, right = m - 1;
5      int result = -1;
6      while (left <= right) {
7          int mid = left + (right - left) / 2;
8          if (arr[mid] == 0) {
9              result = mid;
10             right = mid - 1;
11         } else {
12             left = mid + 1;
13         }
14     }
15     if (result == -1) {
16         return 0;
17     }
18     return m - result;
19 }
20
21 int main() {
22     int m;
23     scanf("%d", &m);
24     int arr[m];
25     for (int i = 0; i < m; i++) {
26         scanf("%d", &arr[i]);
27     }
28     int result = countZeroes(arr, m);
29     printf("%d\n", result);
30     return 0;
31 }
```

	Input	Expected	Got	
✓	5	2	2	✓
	1			
	1			
	1			
	0			
	0			
✓	10	0	0	✓
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
✓	8	8	8	✓
	0			
	0			
	0			
	0			
	0			
	0			
	0			
	0			
✓	17	2	2	✓
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	0			
	0			

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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 2 3	3
7 2 2 1 1 1 2 2	2

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2
3 int largest(int nums[], int low, int high) {
4     if (low == high) {
5         return nums[low];
6     }
7     int mid = low + (high - low) / 2;
8     int left_major = largest(nums, low, mid);
9     int right_major = largest(nums, mid + 1, high);
10    if (left_major == right_major) {
11        return left_major;
12    }
13    int left_count = 0;
14    int right_count = 0;
15    for (int i = low; i <= high; i++) {
16        if (nums[i] == left_major) left_count++;
17        if (nums[i] == right_major) right_count++;
18    }
19    return (left_count > right_count) ? left_major : right_major;
20 }
21
22 int majority_element(int nums[], int n) {
23     return largest(nums, 0, n - 1);
24 }
25
26 int main() {
27     int n;
28     scanf("%d", &n);
29     int nums[n];
30     for (int i = 0; i < n; i++) {
31         scanf("%d", &nums[i]);
32     }
33     printf("%d", majority_element(nums, n));
34     return 0;
35 }
```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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 %)

```
1 #include <stdio.h>
2
3 int findFloor(int arr[], int n, int x) {
4     int left = 0, right = n - 1;
5     int floor = -1;
6     while (left <= right) {
7         int mid = left + (right - left) / 2;
8         if (arr[mid] == x) {
9             return arr[mid];
10        } else if (arr[mid] < x) {
11            floor = arr[mid];
12            left = mid + 1;
13        } else {
14            right = mid - 1;
15        }
16    }
17    return floor;
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     int floorValue = findFloor(arr, n, x);
29     printf("%d\n", floorValue);
30     return 0;
31 }
```

	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! ✓

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 %)

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

	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! ✓

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	12 34 67 78 98
67 34 12 98 78	

Answer:

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