

Problem 1: Divide And Conquer Method

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.

Program:

```
1 #include <stdio.h>
2
3 int countZeros(int arr[], int low, int high, int size) {
4     if (low > high)
5         return 0;
6
7     int mid = (low + high) / 2;
8
9     if (arr[mid] == 1) {
10         return countZeros(arr, mid + 1, high, size);
11     } else {
12
13         if (mid == 0 || arr[mid - 1] == 1) {
14             return size - mid;
15         } else {
16             return countZeros(arr, low, mid - 1, size);
17         }
18     }
19 }
20
21 int main() {
22     int m;
23     scanf("%d", &m);
24
25     int arr[m];
26     for (int i = 0; i < m; i++) {
27         scanf("%d", &arr[i]);
28     }
29
30     int result = countZeros(arr, 0, m - 1, m);
31     printf("%d\n", result);
32
33     return 0;
34 }
```

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

Problem 2: Divide And Conquer Method

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  #include <stdio.h>
2
3  int majorityElement(int* nums, int numsSize) {
4      int count = 0;
5      int candidate = 0;
6
7      for (int i = 0; i < numsSize; i++) {
8          if (count == 0) {
9              candidate = nums[i];
10             count = 1;
11         } else if (nums[i] == candidate) {
12             count++;
13         } else {
14             count--;
15         }
16     }
17     return candidate;
18 }
19
20 int main() {
21     int n;
22     scanf("%d", &n);
23
24     int nums[n];
25     for (int i = 0; i < n; i++) {
26         scanf("%d", &nums[i]);
27     }
28
29     int result = majorityElement(nums, n);
30     printf("%d\n", result);
31
32     return 0;
33 }
34

```

	Input	Expected	Got	
✓	3	3	3	✓
	3 2 3			

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Problem 3: Divide And Conquer Method

Question:

Convert the following algorithm into a program and find its time complexity using counter method.

```
Factor(num) {  
  {  
    for (i = 1; i <= num; ++i)  
    {  
      if (num % i == 0)  
      {  
        printf("%d ", i);  
      }  
    }  
  }  
}
```

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```
1 #include <stdio.h>  
2  
3 int main() {  
4     int num;  
5     scanf("%d", &num);  
6  
7     int counter = 0;  
8     int factor_count = 0;  
9  
10    for (int i = 1; i <= num; ++i) {  
11        counter++;  
12  
13        counter++;  
14        if (num%i == 0){  
15            factor_count++;  
16        }  
17    }  
18    counter += factor_count;  
19    counter++;  
20  
21    printf("%d\n", counter);  
22    return 0;  
23 }  
24  
25
```

	Input	Expected	Got	
✓	12	31	31	✓
✓	25	54	54	✓
✓	4	12	12	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Problem 4: Divide And Conquer Method

Question:

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")

Program:

```

1 #include <stdio.h>
2
3 int findPair(int arr[], int low, int high, int x) {
4     if (low >= high)
5         return 0;
6     int sum = arr[low] + arr[high];
7     if (sum == x) {
8         printf("%d\n", arr[low]);
9         printf("%d\n", arr[high]);
10        return 1;
11    }
12    else if (sum < x)
13        return findPair(arr, low + 1, high, x);
14    else
15        return findPair(arr, low, high - 1, x);
16 }
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     if (!findPair(arr, 0, n - 1, x)) {
27         printf("No\n");
28     }
29     return 0;
30 }
31

```

	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.

Problem 5: Divide And Conquer Method

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

Program:


```

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
13     for (int j = low; j < high; j++) {
14         if (arr[j] <= pivot) {
15             i++;
16             swap(&arr[i], &arr[j]);
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     int arr[n];
35

```

```

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     printf("\n");
46
47     return 0;
48 }
49
50

```

	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.