Started on	Tuesday, 1 October 2024, 1:53 PM
State	Finished
Completed on	Wednesday, 2 October 2024, 12:30 PM
Time taken	22 hours 36 mins
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 %)

```c
#include <stdio.h>

// Function to count zeros using Divide and Conquer
int countZeros(int arr[], int left, int right) {
    // Base case: If there's only one element
    if (left == right) {
        return arr[left] == 0 ? 1 : 0;
    }

    // Find the middle index
    int mid = left + (right - left) / 2;

    // Count zeros in the left half and the right half
    int leftCount = countZeros(arr, left, mid);
    int rightCount = countZeros(arr, mid + 1, right);

    // Combine the counts
    return leftCount + rightCount;
}

int main() {
    int m;
    scanf("%d", &m); // Read the size of the array

    int arr[m]; // Declare the array

    // Read the array elements
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }

    // Count the number of zeros
    int zeroCount = countZeros(arr, 0, m - 1);

    // Print the result
    printf("%d\n", zeroCount);

    return 0;
}
```

	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.

◄ Problem 5: Finding Complexity using counter method

Jump to...

2-Majority Element ►

Started on	Wednesday, 2 October 2024, 1:04 PM
State	Finished
Completed on	Wednesday, 2 October 2024, 1:10 PM
Time taken	6 mins 5 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int majorityElement(int* nums, int numsSize) {
    int candidate = 0;
    int count = 0;

    for (int i = 0; i < numsSize; i++) {
        if (count == 0) {
            candidate = nums[i];
            count = 1;
        } else if (nums[i] == candidate) {
            count++;
        } else {
            count--;
        }
    }

    return candidate;
}

int main() {
    int n;
    scanf("%d", &n);

    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
```

```
    }
    int majority = majorityElement(nums, n);
    printf("%d\n", majority);

    return 0;
}
```

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 1-Number of Zeros in a Given Array

Jump to...

3-Finding Floor Value ►

Started on	Wednesday, 2 October 2024, 1:10 PM
State	Finished
Completed on	Wednesday, 2 October 2024, 1:13 PM
Time taken	2 mins 27 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 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 %)

```c
#include <stdio.h>

int findFloor(int* arr, int n, int x) {
    int left = 0;
    int right = n - 1;
    int floorValue = -1; // Default value if no floor is found

    while (left <= right) {
        int mid = left + (right - left) / 2;

        // If we find an exact match
        if (arr[mid] == x) {
            return arr[mid];
        }
        // If the middle element is less than or equal to x
        else if (arr[mid] < x) {
            floorValue = arr[mid]; // Update floorValue
            left = mid + 1; // Move right
        }
        // If the middle element is greater than x
        else {
            right = mid - 1; // Move left
        }
    }
    return floorValue; // Return the largest element <= x
}

int main() {
    int n;


    scanf("%d", &n);

    int arr[n];

    // Input elements of the array

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Input value of x
    int x;

    scanf("%d", &x);

    // Find and print the floor value of x
    int floorValue = findFloor(arr, n, x);
    printf("%d\n",floorValue); // Output the floor value
```

```
51      return 0;
52  }
```

	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.

◄ 2-Majority Element

Jump to...

4-Two Elements sum to x ►

Started on	Wednesday, 2 October 2024, 1:13 PM
State	Finished
Completed on	Wednesday, 2 October 2024, 1:16 PM
Time taken	2 mins 45 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 findPairWithSum(int* arr, int left, int right, int x) {
    if (left >= right) {
        printf("No\n");
        return;
    }

    int currentSum = arr[left] + arr[right];
    if (currentSum == x) {
        printf("%d\n", arr[left]);
        printf("%d\n", arr[right]);
        return;
    } else if (currentSum < x) {
        findPairWithSum(arr, left + 1, right, x);
    } else {
        findPairWithSum(arr, left, right - 1, x);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int x;
    scanf("%d", &x);
    findPairWithSum(arr, 0, n - 1, x);

    return 0;
}

```

	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔

	Input	Expected	Got	
✔	5 2 4 6 8 10 100	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 3-Finding Floor Value

Jump to...

6-Implementation of Quick Sort ►

Started on	Wednesday, 2 October 2024, 1:16 PM
State	Finished
Completed on	Wednesday, 2 October 2024, 1:18 PM
Time taken	1 min 32 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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:

```c
#include <stdio.h>
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Partition function for Quick Sort
int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // Choosing the last element as the pivot
    int i = (low - 1); // Index of the smaller element

    for (int j = low; j < high; j++) {
        // If the current element is smaller than or equal to the pivot
        if (arr[j] <= pivot) {
            i++; // Increment index of smaller element
            swap(&arr[i], &arr[j]); // Swap
        }
    }
    swap(&arr[i + 1], &arr[high]); // Swap the pivot element with the element at i+1
    return (i + 1); // Return the partition index
}

// Quick Sort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // Partitioning index
        int pi = partition(arr, low, high);

        // Recursively sort elements before and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int n;


    scanf("%d", &n);

    int arr[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
```

```
46      }
47
48      // Perform Quick Sort
49      quickSort(arr, 0, n - 1);
50
51      // Output sorted elements
52
```

	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.

◄ 4-Two Elements sum to x

Jump to...

1-G-Coin Problem ►