Started on	Thursday, 19 September 2024, 8:58 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:46 AM
Time taken	13 days 23 hours
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>

int countZeroes(int arr[], int left, int right) {
    if (left > right) {
        return 0;
    }
    if (left == right) {
        return arr[left] == 0 ? 1 : 0;
    }

    int mid = left + (right - left) / 2;
    return countZeroes(arr, left, mid) + countZeroes(arr, mid + 1, right);
}

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

    int zeroCount = countZeroes(arr, 0, m - 1);
    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	✔

	Input	Expected	Got	
✔	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 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	Thursday, 3 October 2024, 8:45 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:46 AM
Time taken	41 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 %)

```
#include <stdio.h>
int majorityElement(int* n
int candidate = nums[0];
int count = 1;
for (int i = 1; i < numsSi
if (nums[i] == candidate)
count++;
} else {
count--;
if (count == 0) {
candidate = nums[i];
count = 1;
}
}
}
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 = majorityEle
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	Thursday, 3 October 2024, 8:46 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:47 AM
Time taken	30 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, right = n - 1, floorIndex = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            floorIndex = mid;
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return (floorIndex != -1) ? arr[floorIndex] : -1;
}
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);
    int floorValue = findFloor(arr, n, x);
    printf("%d\n", floorValue);
    return 0;
}
```

	Input	Expected	Got	
✔	6 1 2 8 10 12 19 5	2	2	✔
✔	5 10 22 85 108 129 100	85	85	✔

	Input	Expected	Got	
✔	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	Thursday, 3 October 2024, 8:47 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:48 AM
Time taken	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 findPair(int arr[], int left, int right, int x) {
    while (left < right) {
        int sum = arr[left] + arr[right];
        if (sum == x) {
            printf("%d\n", arr[left]);
            printf("%d\n", arr[right]);
            return;
        } else if (sum < x) {
            left++;
        } else {
            right--;
        }
    }
    printf("No\n");
}

void checkPair(int arr[], int n, int x) {
    findPair(arr, 0, n - 1, x);
}

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

	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.

◄ 3-Finding Floor Value

Jump to...

6-Implementation of Quick Sort ►

Started on	Thursday, 3 October 2024, 8:48 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:48 AM
Time taken	36 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 quickSort(int arr[], int low, int high);
int partition(int arr[], int low, int high);
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d ", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

```

	Input	Expected	Got	
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✔

	Input	Expected	Got	
✔	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 ►