(1) **Question 1**

Correct

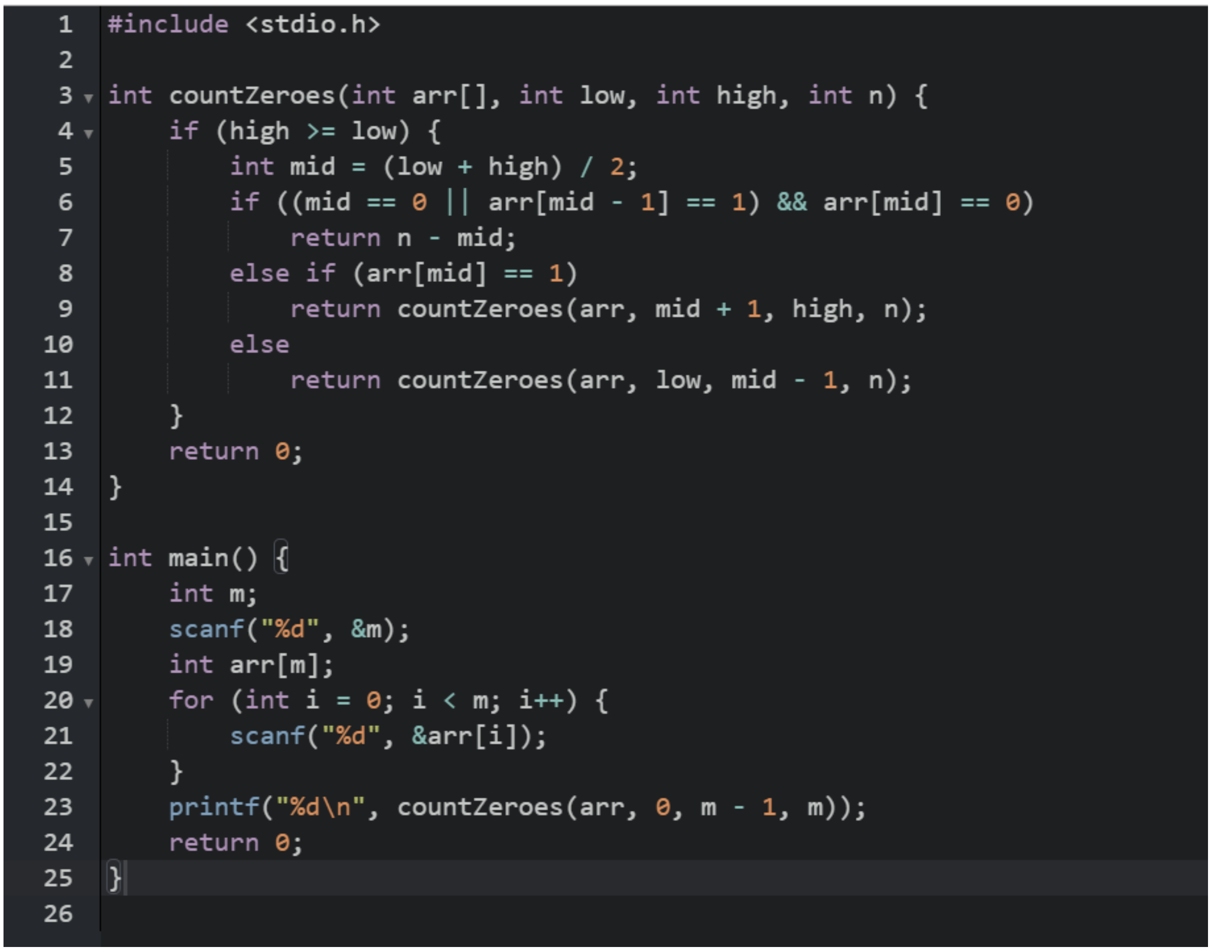
Mark 1.00 out of 1.00

Flag question

Question text

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 %)

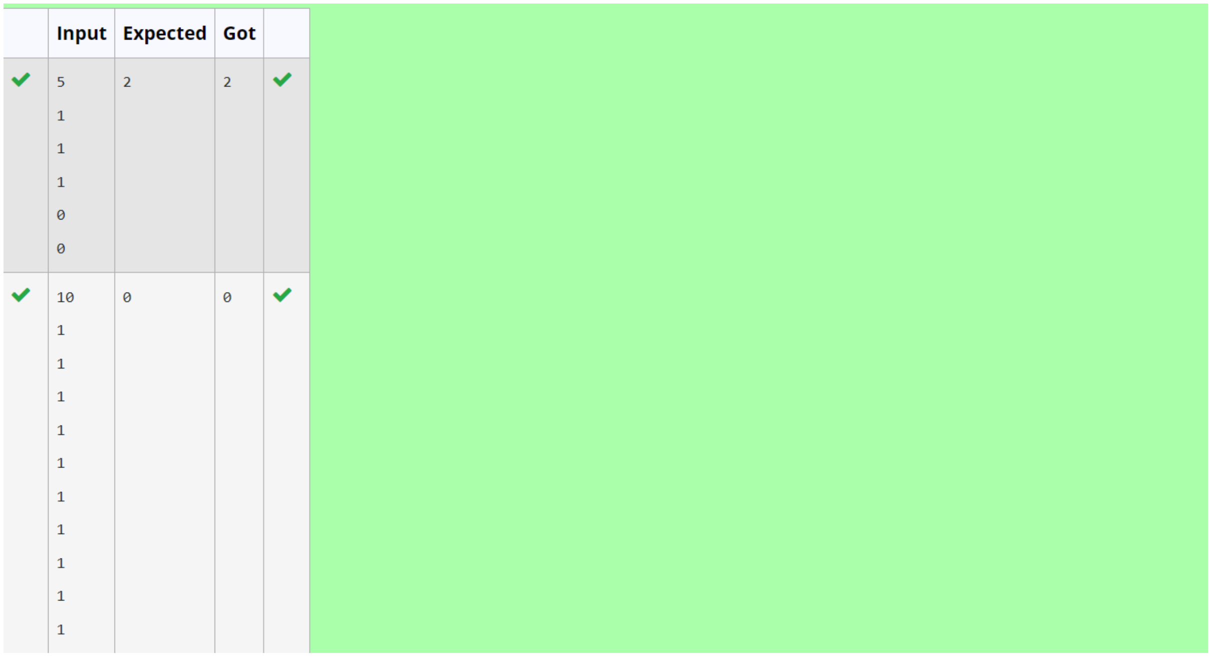


```c
#include <stdio.h>

int countZeroes(int arr[], int low, int high, int n) {
    if (high >= low) {
        int mid = (low + high) / 2;
        if ((mid == 0 || arr[mid - 1] == 1) && arr[mid] == 0)
            return n - mid;
        else if (arr[mid] == 1)
            return countZeroes(arr, mid + 1, high, n);
        else
            return countZeroes(arr, low, mid - 1, n);
    }
    return 0;
}

int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    printf("%d\n", countZeroes(arr, 0, m - 1, m));
    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	✓

(2) **2-Majority Element**

Started on	Thursday, 25 September 2025, 10:16 PM
State	Finished
Completed on	Thursday, 25 September 2025, 10:17 PM
Time taken	35 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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:

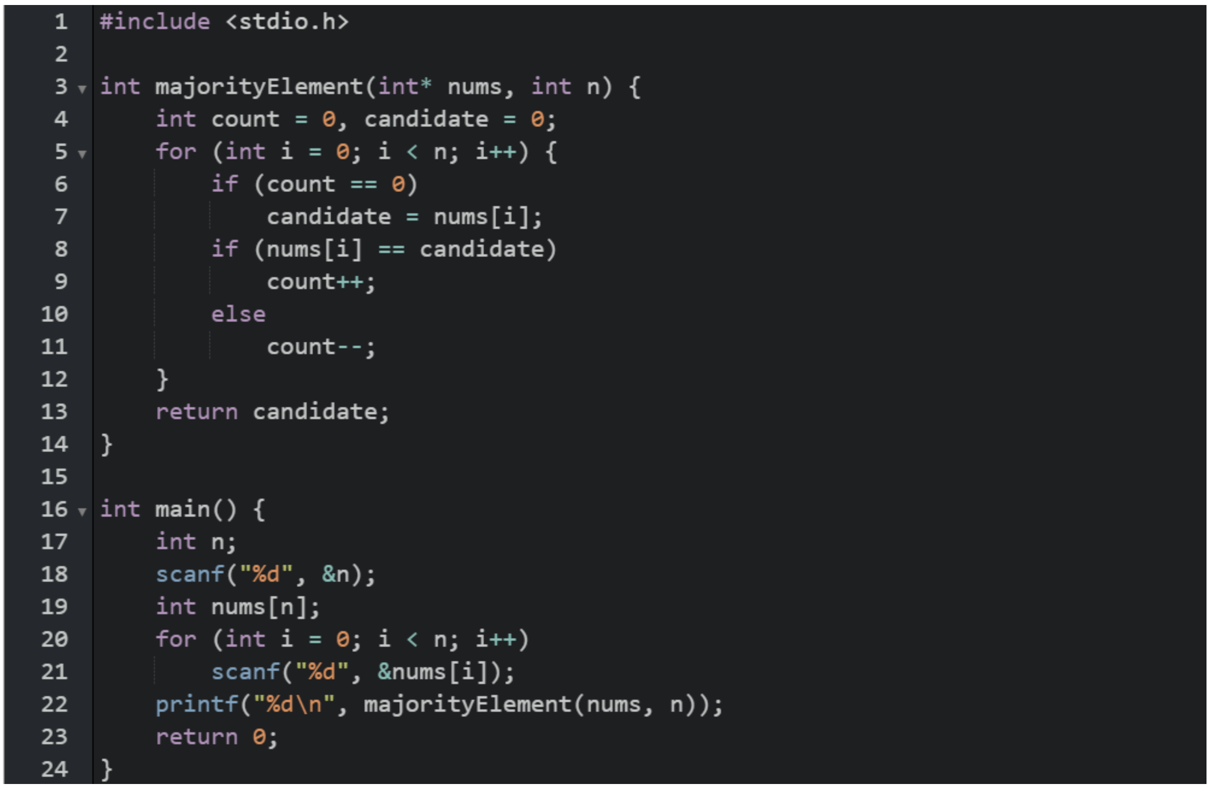
- n == nums.length

- $1 <= n <= 5 * 10^4$
- $-2^{31} <= nums[i] <= 2^{31} - 1$

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 n) {
    int count = 0, candidate = 0;
    for (int i = 0; i < n; i++) {
        if (count == 0)
            candidate = nums[i];
        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]);
    printf("%d\n", majorityElement(nums, n));
    return 0;
}
```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

(3) **3-Finding Floor Value**

Started on	Thursday, 25 September 2025, 10:17 PM
State	Finished
Completed on	Thursday, 25 September 2025, 10:18 PM
Time taken	38 secs

Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Question text

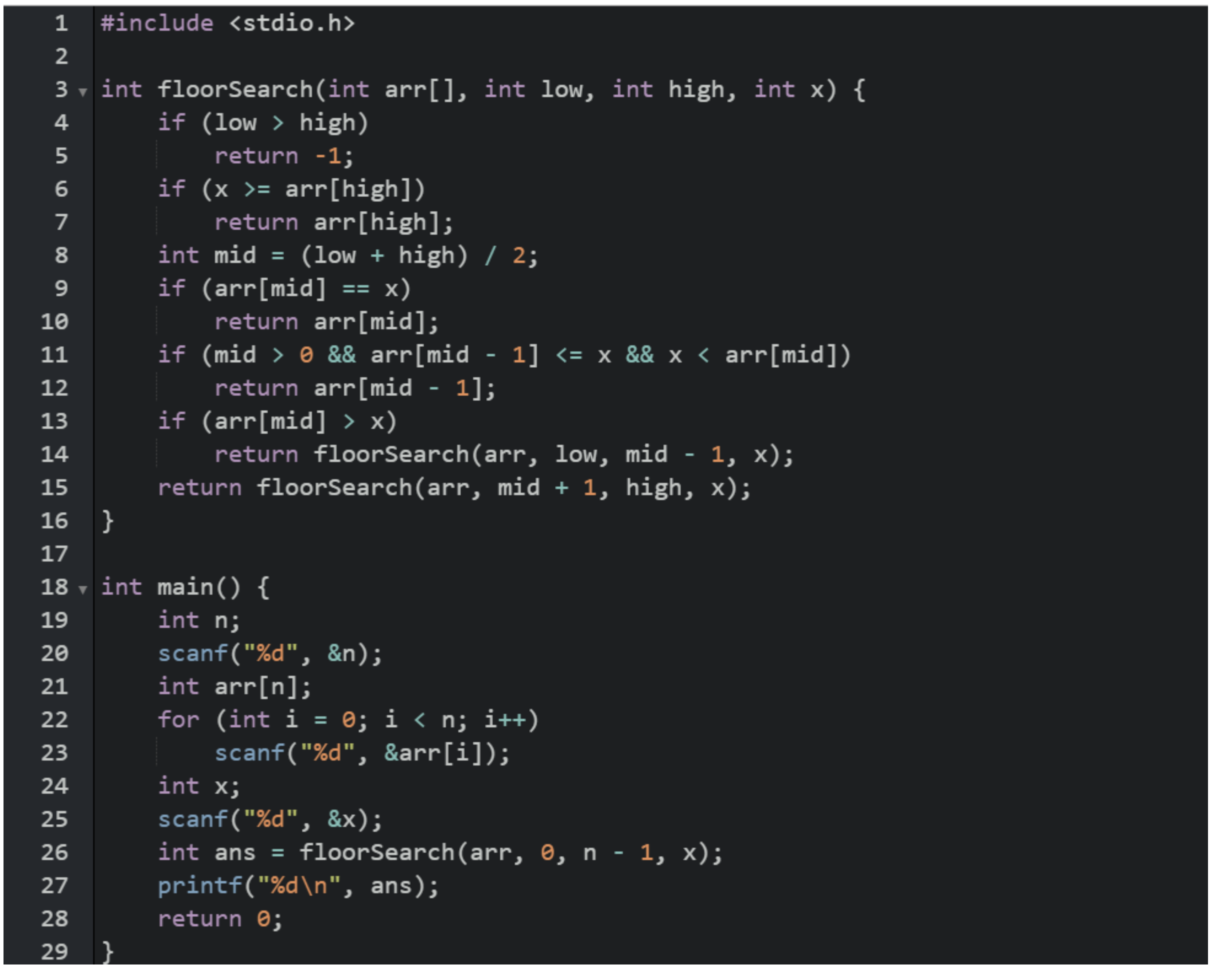
Problem Statement:
Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

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

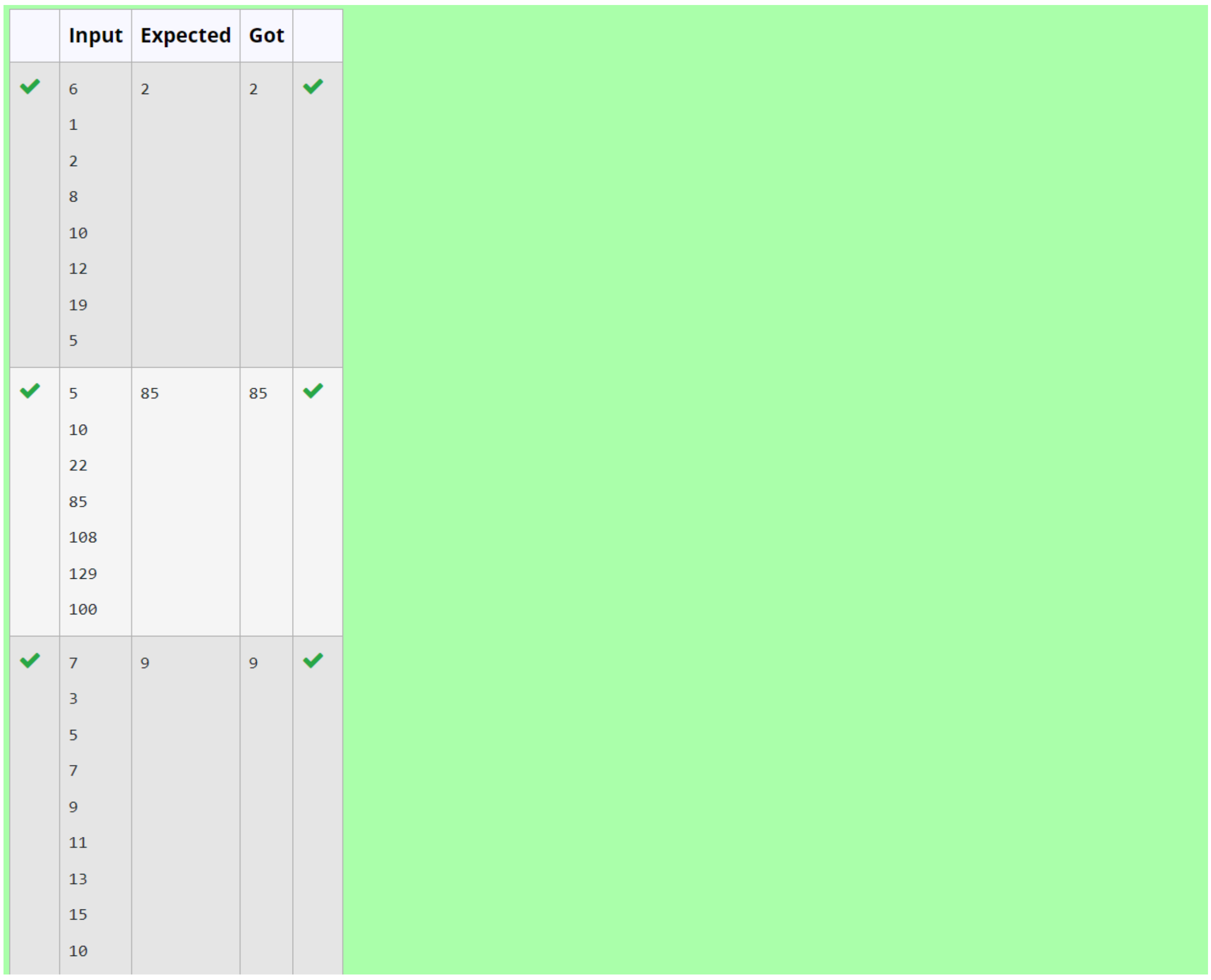
First Line Contains Integer – Floor value for x



```c
#include <stdio.h>

int floorSearch(int arr[], int low, int high, int x) {
    if (low > high)
        return -1;
    if (x >= arr[high])
        return arr[high];
    int mid = (low + high) / 2;
    if (arr[mid] == x)
        return arr[mid];
    if (mid > 0 && arr[mid - 1] <= x && x < arr[mid])
        return arr[mid - 1];
    if (arr[mid] > x)
        return floorSearch(arr, low, mid - 1, x);
    return floorSearch(arr, mid + 1, high, 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);
    int ans = floorSearch(arr, 0, n - 1, x);
    printf("%d\n", ans);
    return 0;
}
```



	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	✓

(4) **4-Two Elements sum to x**

Started on	Thursday, 25 September 2025, 10:18 PM
State	Finished
Completed on	Thursday, 25 September 2025, 10:18 PM
Time taken	40 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

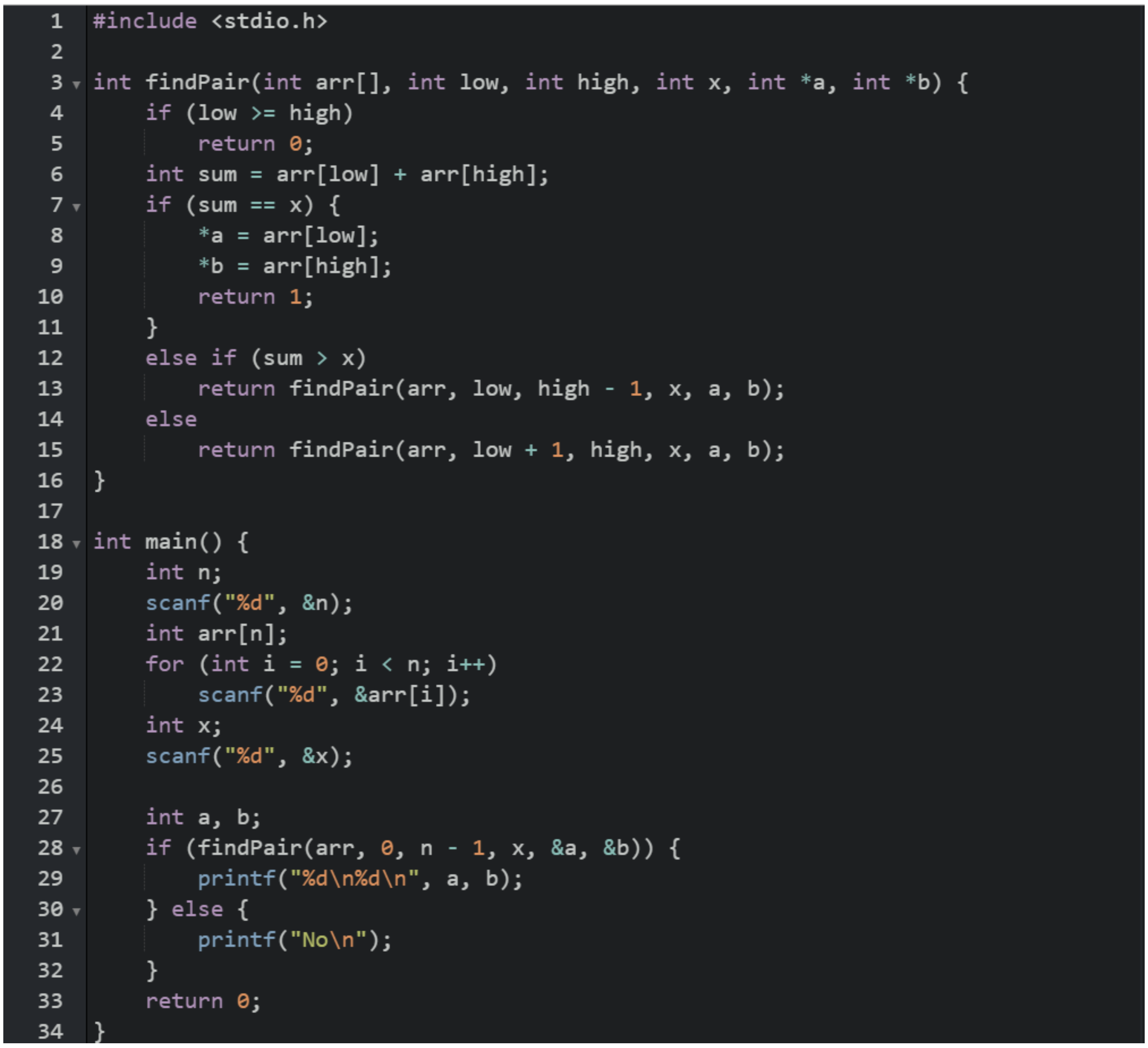
Mark 1.00 out of 1.00

Flag question

Question text

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>

int findPair(int arr[], int low, int high, int x, int *a, int *b) {
    if (low >= high)
        return 0;
    int sum = arr[low] + arr[high];
    if (sum == x) {
        *a = arr[low];
        *b = arr[high];
        return 1;
    }
    else if (sum > x)
        return findPair(arr, low, high - 1, x, a, b);
    else
        return findPair(arr, low + 1, high, x, a, b);
}

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 a, b;
    if (findPair(arr, 0, n - 1, x, &a, &b)) {
        printf("%d\n%d\n", a, b);
    } else {
        printf("No\n");
    }
    return 0;
}
```

	Input	Expected	Got	
✓	4 2 4 8 10 14	4 10	4 10	✓
✓	5 2 4 6 8 10 100	No	No	✓

(5) **5-Implementation of Quick Sort**

Started on	Thursday, 25 September 2025, 10:19 PM
State	Finished
Completed on	Thursday, 25 September 2025, 10:20 PM
Time taken	44 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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.

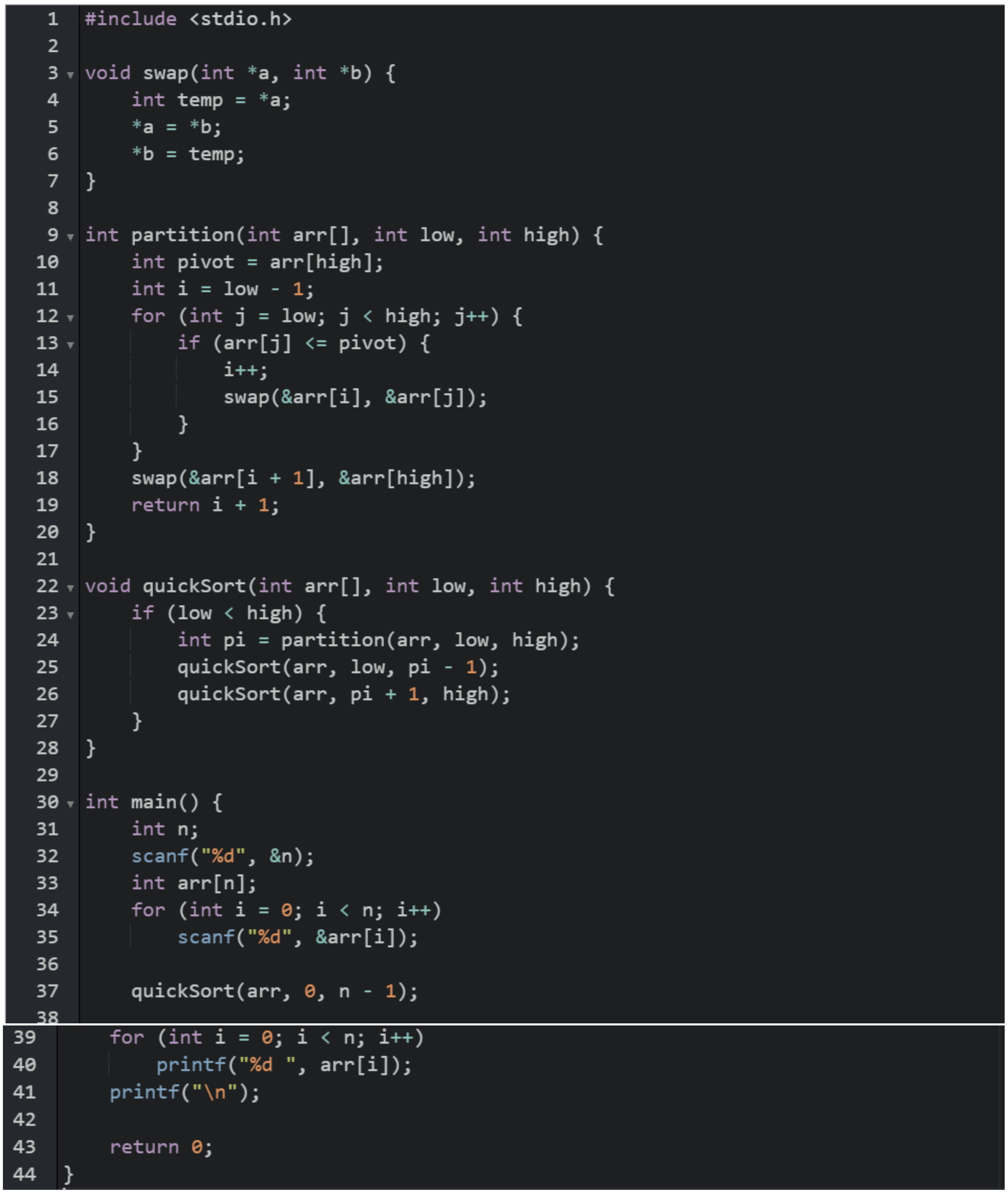
Output:
Sorted list of elements

For example:

Input	Result
5	12 34 67 78 98

Input	Result
67 34 12 98 78	

Answer:



```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

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++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}

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 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]);
    printf("\n");

    return 0;
}
```

	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! ✓