Name	Aravinthaa S
Register No	230701032
Dept	CSE A

DIVIDE AND CONQUER

Ex No – 3.1	NUMBER OF ZEROES IN A GIVEN ARRAY

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.

Aim:

To count the number of 0s in a sorted array of 1s and 0s using the Divide and Conquer technique

Program :

```
#include <stdio.h>
int countZeroes(int arr[], int low, int high) {
    if (low == high) {
        return arr[low] == 0 ? 1 : 0;
    }
    int mid = (low + high) / 2;
    int leftZeroCount = countZeroes(arr, low,
mid);
    int rightZeroCount = countZeroes(arr,
mid + 1, high);
    return leftZeroCount + rightZeroCount;
}
int main() {
    int m;
    scanf("%d", &m);
```

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

Input and Output:

	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 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! ✔

Ex No – 3.2	MAJORITY ELEMENT

Question:

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

Aim:

To find the majority element in an array, which appears more than ⌊n / 2⌋ times, using an efficient algorithm.

Program:

```
#include <stdio.h>


int majorityElement(int nums[], int size) {
    int candidate = nums[0];
    int count = 1;


    for (int i = 1; i < size; 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 result = majorityElement(nums, n);
    printf("%d\n", result);

    return 0;
}
```

Input and Output:

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Ex No – 3.3	FINDING FLOOR VALUE

Question:

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

Aim:

To find the floor of a given value x in a sorted array using a Divide and Conquer algorithm.

Program:

```
#include <stdio.h>


int findFloor(int arr[], int low, int high, int x) {
    int result = -1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] <= x) {
            result = arr[mid];
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return result;
}


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, 0, n - 1, x);
    printf("%d\n", floorValue);

    return 0;
}
```

Input amd Output:

	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! ✔

Ex No – 3.4	TWO ELEMENTS SUM TO X

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")

Aim:

To check if there exist two elements in a sorted array whose sum equals a given value x using a Divide and Conquer approach.

Program :

```
#include <stdio.h>


void findPairRecursive(int arr[], int left, int right, int x) {
    if (left >= right) {
        printf("No\n");
        return;
    }

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

    findPairRecursive(arr, 0, n - 1, x);

    return 0;
}
```

Input and Output:

	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! ✔

Ex No – 3.5	IMPLEMENTATION OF QUICK SORT

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

Aim:

To implement the Quick Sort algorithm to sort a list of elements efficiently.

Program:

```
#include <stdio.h>


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;
}

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++) {
        if (i > 0) {
```

```
            printf(" ");
        }
        printf("%d", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Input and Output:

	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! ✔