DAA MOODLE PROGRAMS
DIVIDE AND CONQUER PROGRAMS

230701037
Arun Prakash M
CSE-A

1.

AIM-

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.

CODE-

```c
#include <stdio.h>
int findFirstZero(int arr[], int low, int high) {
    if (low > high)
    return -1;
    int mid = low + (high - low) / 2;
    if (arr[mid] == 0 && (mid == 0 || arr[mid - 1] == 1))
    return mid;
    return arr[mid] == 1 ? findFirstZero(arr, mid + 1, high) : findFirstZero(arr, low, mid - 1);
}

int countZeroes(int arr[], int size) {
    int firstZero = findFirstZero(arr, 0, size - 1);
    return firstZero == -1 ? 0 : size - firstZero;
}

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, m));
    return 0;
}
```

INPUT-

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

OUTPUT-

First Line Contains Integer – Number of zeroes present in the given array.

	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! ✔

2.

AIM-

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.

CODE-

```c
#include <stdio.h>
int find(int nums[], int n) {
    int count = 0;
    int t = 0;

    for (int i = 0; i < n; i++)
    {
        if (count == 0) {
            t = nums[i];
        }
        count += (nums[i] == t) ? 1 : -1;
    }
    return t;
}

int main() {
    int n;
    scanf("%d", &n);
    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    int majele = find(nums, n);
    printf("%d\n", majele);
    return 0;
}

```

INPUT-

```
nums = [3,2,3]
```

OUTPUT-
3

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.
AIM-

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.

CODE-

```c
#include <stdio.h>
int findFloor(int arr[], int low, int high, int x) {
    if (x < arr[low])
    return -1;
    if (x >= arr[high]) return arr[high];
    int mid = low + (high - low) / 2;
    if (arr[mid] == x) {
        return arr[mid];
    }
    if (arr[mid] < x) {
        if (mid + 1 <= high && arr[mid + 1] > x) {
            return arr[mid];
        }
        return findFloor(arr, mid + 1, high, x);
    }
    return findFloor(arr, low, mid - 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);
    int floor = findFloor(arr, 0, n - 1, x);
    if (floor == -1) {
        printf("No floor found\n");
    } else {
        printf("%d\n", floor);
    }

    return 0;
}
```

INPUT-

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-

First Line Contains Integer – Floor value for x

	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.

4.

AIM-

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

CODE-

```c
#include <stdio.h>
void findpair(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]);
        return;
    } else if (sum < x) {
        findpair(arr, left + 1, right, x);
    } else {
        findpair(arr, left, right - 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);
    findpair(arr, 0, n - 1, x);
    return 0;
}
```

INPUT-

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-

First Line Contains Integer – Element1

Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

	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.

5.

AIM-

Write a Program to Implement the Quick Sort Algorithm.

CODE-

```c
#include <stdio.h>
void swap(int arr[], int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[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, j);
        }
    }
    swap(arr, i + 1, 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-

The first line contains the no of elements in the list-n
The next n lines contain the elements.

OUTPUT-

Sorted list of elements

	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.