ASSIGNMENT 2

Q.NO 1

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

ANSWER

```
#include <stdio.h>
int majority(int nums[], int low, int high)
{
   if (low==high)
      return nums[low];
   int mid=(low+high)/2;
   int left=majority(nums,low, mid);
   int right=majority(nums, mid + 1,high);
   if (left==right)
      return left;
   int lc=0;
   for (int i=low;i<=high;i++)
```

```
        if (nums[i] == left)
            lc++;
    int rc=0;
    for (int i=low;i<=high;i++)
        if (nums[i]==right)
            rc++;
    if (lc>(low-high+1)/2)
        return left;
    if (rc>(low-high+1)/2)
        return right;
    return -1;
}
int main()
{
int n;
scanf("%d",&n);
int nums[n];
for(int i=0;i<n;i++)
    scanf("%d",&nums[i]);
printf("%d",majority(nums,0,n-1));
return 0;
}
```

Input	Expected	Got
3 3 2 3	3	3

Q.NO 2

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

```
#include<stdio.h>
int fV(int arr[], int lo, int h, int x)
{
   if (lo>h)
      return 0;
   int mid=lo+(h-lo)/2;
   if (arr[mid]==x)
      return arr[mid];
   else if (arr[mid]<x) {
      int f=fV(arr,mid+1,h,x);
      if(f==0)
         return arr[mid];
      else
         return f;
   }
```

```
    else
        return fV(arr,lo,mid-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);
    printf("%d",fV(arr,0,n-1,x));
}
```

Input	Expected	Got
6 1 2 8 10 12 19 5	2	2