DIVIDE AND CONQUER **M.MOHANA**

231901031

1.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.

Solution:

```
#include<stdio.h>

int countzeroes(int arr[],int low,int high,int n)

{

  if(low>high)

  {

    return 0;

  }

  int mid=(low+high)/2;

  if(arr[mid]==0)

  {

    if(mid==0||arr[mid-1]==1)

     return n-mid;

    else

     return countzeroes(arr,low,mid-1,n);

  }

  else

  {

    return countzeroes(arr,mid+1,high,n);

  }
```

```
}

int main()

{

int n;

scanf("%d",&n);

int arr[n];

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

int zeroes=countzeroes(arr,0,n-1,n);

printf("%d",zeroes);

return 0;

}
```

2. 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`

Solution:

```
#include<stdio.h>
```

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

3.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

Solution:

```
#include<stdio.h>
int findfloor(int arr[],int low,int high,int x)
{
   if(low>high)
   {
      return -1;
   }
   if(x<arr[low])
   {
      return -1;
   }
   if(x>=arr[high])
   {
      return arr[high];
   }
   int mid=low+high/2;
   if(arr[mid]==x)
   {
      return arr[mid];
   }
   if(mid<high && arr[mid]<x &&arr[mid+1]>x)
   {
      return arr[mid];
   }
   if(x<arr[mid])
   {
     return findfloor(arr,low,mid-1,x);
   }
   return findfloor(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 floorvalue=findfloor(arr,0,n-1,x);
   printf("%d",floorvalue);
   return 0;

}
```

4.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")

Solution:

```
#include<stdio.h>

int binarysearch(int arr[],int low,int high,int x)
```

```
{
if(low>high)
{
   return -1;
}
int mid=(low+high)/2;
if(arr[mid]==x)
{
  return  mid;
}
else if(arr[mid]>x)
{
return binarysearch(arr,low,mid-1,x);
}
else
{
   return binarysearch(arr,mid+1,high,x);
}
}
void twosum(int arr[],int n,int x)
{
for(int i=0;i<n;i++)
{
int complement=x-arr[i];
if(binarysearch(arr,i+1,n-1,complement)!=-1)
{
printf("%d\n%d",arr[i],complement);
```

```
return ;
}
}
printf("No");
}
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);
twosum(arr,n,x);
return 0;
}
```

5. 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

Solution:

```
#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 p=arr[low];
   int i=low;
   int j=high;
   while(i<j)
   {
      while(arr[i]<=p && i<=high-1)
      {
         i++;
      }
      while(arr[j]>p && j>=low+1)
      {
         j--;
      }
      if(i<j)
      {
         swap(&arr[i],&arr[j]);
      }
   }
```

```
    swap(&arr[low],&arr[j]);
    return j;
}
void quick(int arr[],int low,int high)
{
    if(low<high)
    {
        int pi=partition(arr,low,high);
        quick(arr,low,pi-1);
        quick(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]);
    }
    quick(arr,0,n-1);
    for(int i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    return 0;
```

```
}
```