04 - Divide and Conquer

Ex. No. : 4.1 **Date: 03.09.24**

Register No.: 230701513 **Name: R.Veerandira saran**

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.

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array arr of size n. Read n values into the array.

Step 3: Check if the first element of arr is 0. If true, print n and exit the program.

Step 4: Call the divide function with arr, 0, and n-1 to find the index of the first occurrence of 0.

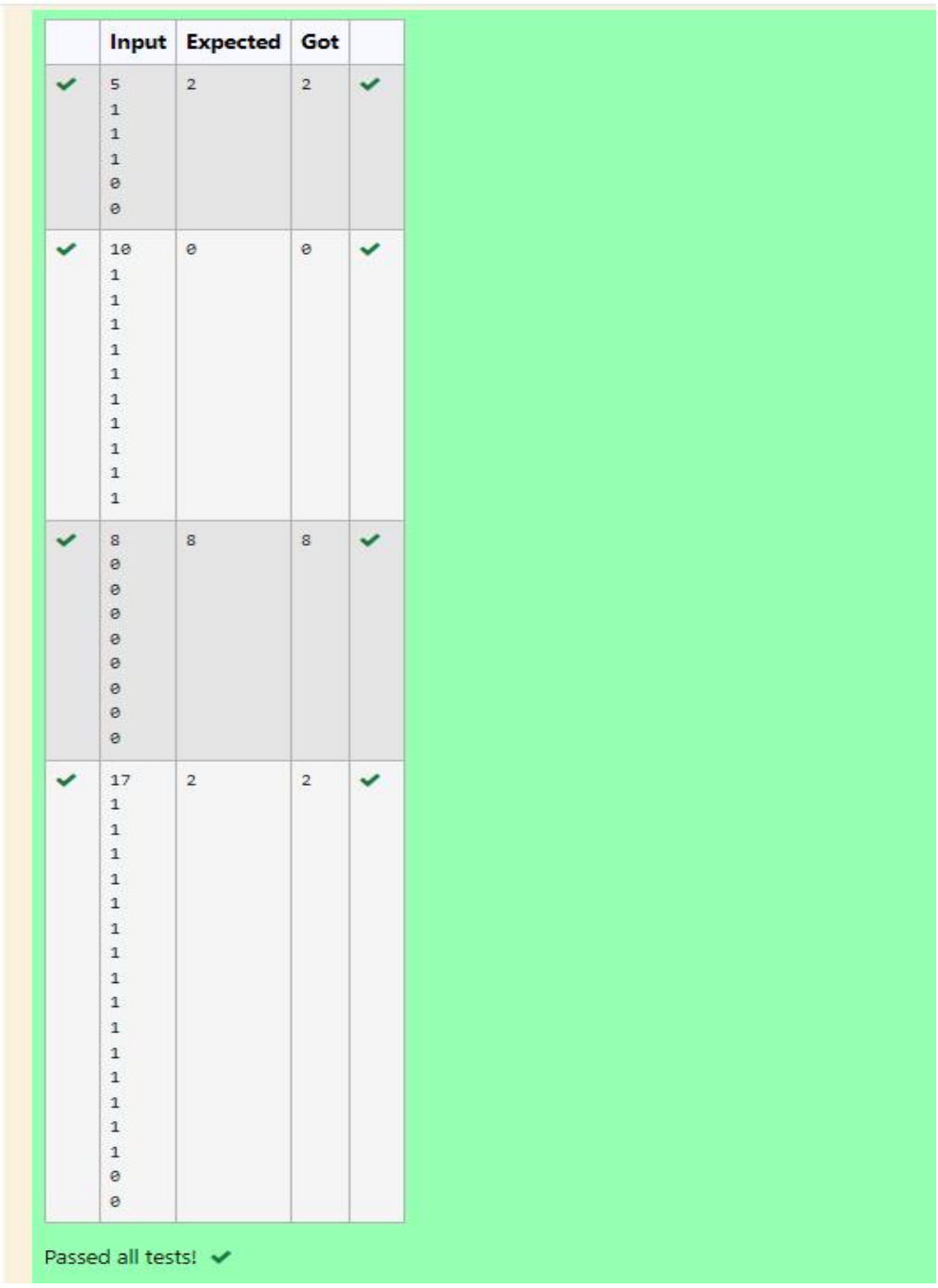
Step 5: If the index is not 0, print the value of n - index, which represents the count of 0s in the array. Otherwise, print the index.

Step 6: End

PROGRAM:

```
#include<stdio.h>
int count=0;
void findCount(int a[],int l,int r){
    if(a[l]==0){
        count+=(r-l+1);
    }else{
        if(l<r){
            int m=(l+r)/2;
            findCount(a,l,m);
            findCount(a,m+1,r);
        }
    }
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    findCount(a,0,n-1);
    printf("%d",count);
}
```


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	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! ✔

RESULT :

Hence the above program has been executed successfully.

Ex. No. : 4.2 **Date: 03.09.24**

Register No.: 230701513 **Name: R.Veerandira saran**

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.

Example 1:

Input: nums = [3,2,3]

Output: 3

Example 2:

Input: nums = [2,2,1,1,1,2,2]

Output: 2

For example:

Input	RESULT
3 3 2 3	3
7 2 2 1 1 1 2 2	2

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array arr of size n. Read n values into the array.

Step 3: Use qsort to sort the array arr in ascending order.

Step 4: Loop through the array to find the first and last indices of each element using the first and last functions. Calculate the count of occurrences (major).

Step 5: If any element's count is greater than or equal to n/2, return that element.

Step 6: Print the element that appears more than n/2 times or print 0 if none is found.

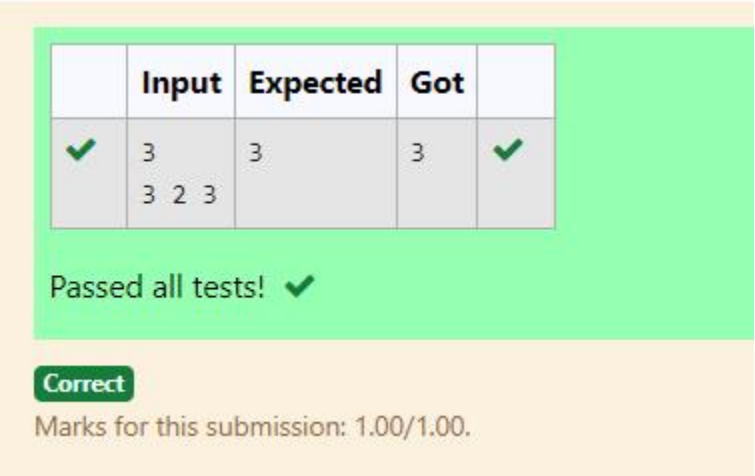
Step 7: End

PROGRAM:

```
#include<stdio.h>

int divide(int a[],int l,int h,int n){

    if(l==h){return a[l];}

    int mid=(l+h)/2;

    int low=divide(a,l,mid,n);

    int high=divide(a,mid+1,h,n);

    int lc=0,rc=0;
```

```
    for(int i=0;i<n;i++){
        if(a[i]==low)
            lc++;
        else
            rc++;
    }
    if(lc>(n/2)){return low;}
    else{return high;}
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    int l=0,h=n-1;
    int m=divide(a,l,h,n);
    printf("%d",m);
}
```


OUTPUT:



	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT :

Hence the above program has been executed successfully.

Ex. No. : 4.3 **Date: 03.09.24**

Register No.: 230701513 **Name: R.Veerandira saran**

AIM:

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array arr of size n. Read n values into the array.

Step 3: Read the integer x from the user, which will be used to find the floor value.

Step 4: Call the search function with arr, x, 0, and n-1 to find the largest element in arr that is less than or equal to x.

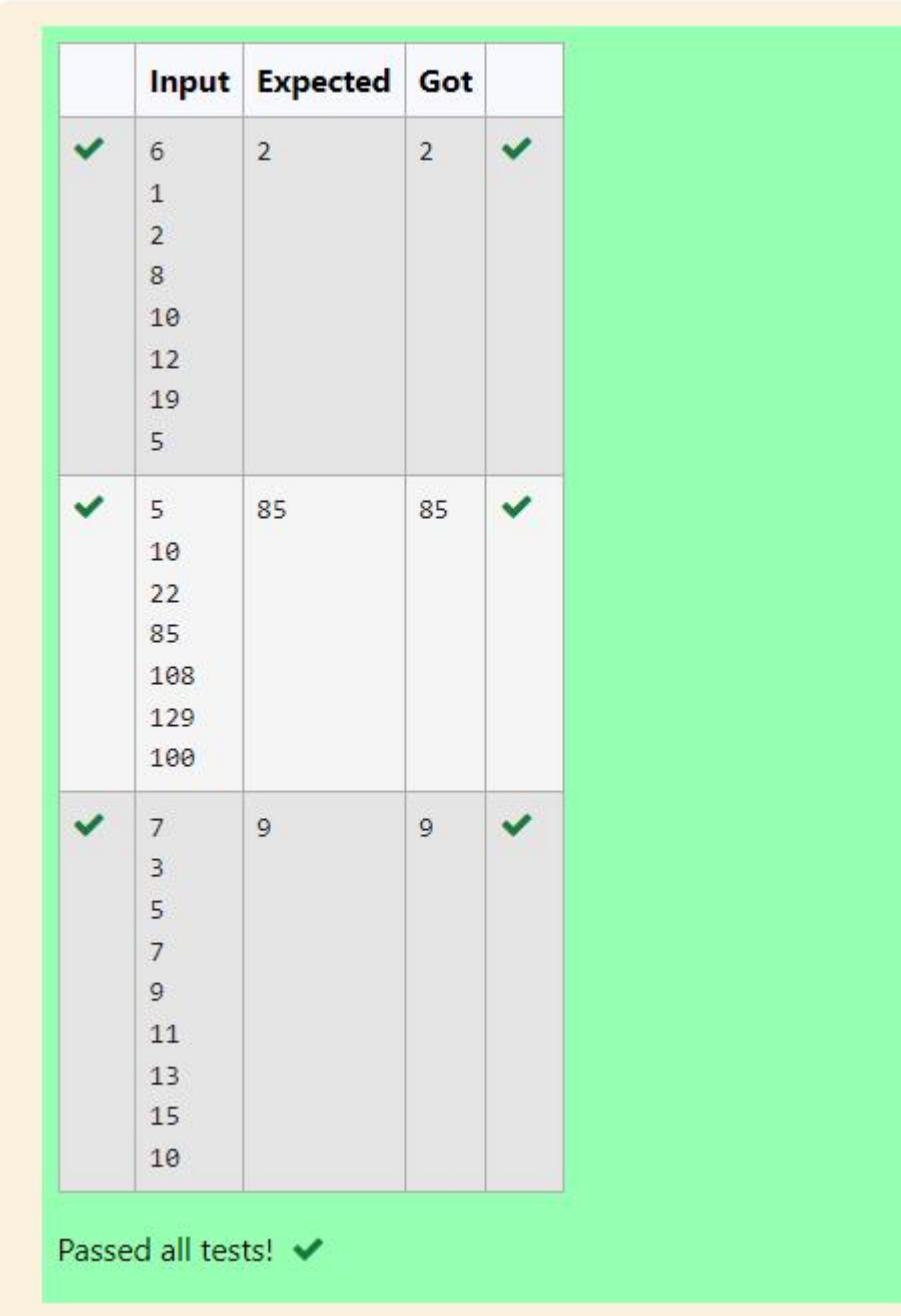
Step 5: Print the floor value returned by the search function.

Step 6: End

PROGRAM:

```
#include <stdio.h>
void findx(int a[],int l,int h,int x){
    if(l<h){
        int mid=(l+h)/2;
        if(a[mid]>x){
            findx(a,l,mid,x);
        }else{
            findx(a,mid+1,h,x);
        }
    }else{
        printf("%d",a[l-1]);
    }
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    int x,l=0;
    scanf("%d",&x);
    findx(a,l,n,x);
}
```

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! ✔

RESULT :

Hence the above program has been executed successfully.

Ex. No. : 4.4 **Date: 03.09.24**

Register No.: 230701513 **Name: R.Veerandira saran**

AIM:

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”)

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array arr of size n. Read n values into the array.

Step 3: Read the integer x from the user, which represents the target sum.

Step 4: Call the twosum function with arr, 0, n-1, and x to find two numbers in the array that add up to x.

Step 5: If a pair is found, print the two numbers; otherwise, print "No" to indicate that no such pair exists.

Step 6: End

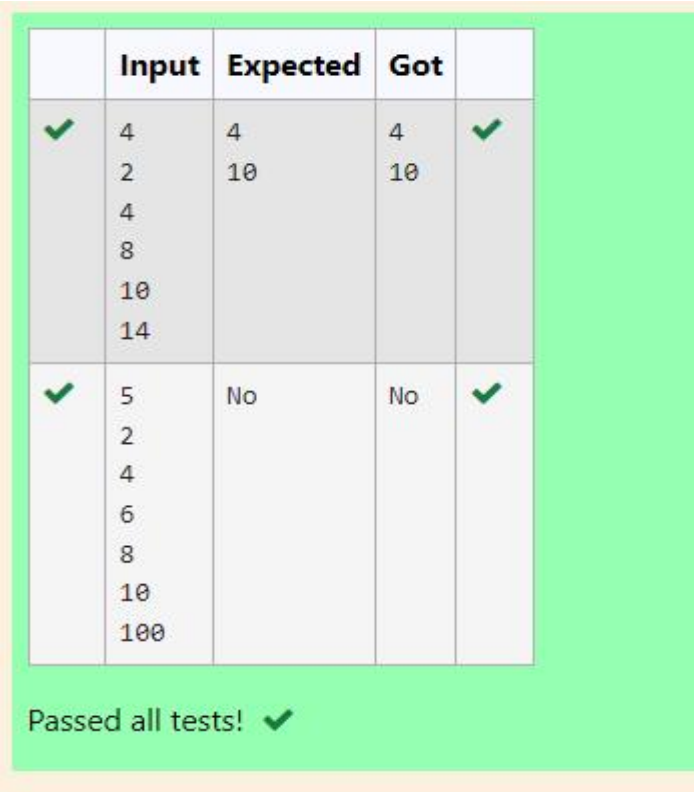
PROGRAM:

```
#include <stdio.h>

int findpair(int arr[],int l,int r,int x){
    if(l>=r){
        return 0;
    }
    int i=l;
    int j=r;
    while(i<j){
```

```
        int sum=arr[i]+arr[j];
        if(sum==x){
            printf("%d\n%d\n",arr[i],arr[j]);
            return 1;
        }else if(sum<x){
            i++;
        }else{
            j--;
        }
    }
    int m=(l+r)/2;
    return findpair(arr,l,m,x)||findpair(arr,m+1,r,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);
    if(!findpair(arr,0,n-1,x)){
        printf("No\n");
    }
}
```

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! ✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 4.5 **Date: 03.09.24**

Register No.: 230701513 **Name: R.Veerandira saran**

AIM:

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

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and dynamically allocate an array arr of size n. Read n values into the array.

Step 3: Call the q_sort function with arr, 0, and n-1 to sort the array using the Quick Sort algorithm.

Step 4: In the q_sort function, select a pivot and partition the array into two halves. Recursively apply the same sorting process to both halves.

Step 5: Once sorted, iterate through the array and print the sorted values.

Step 6: End

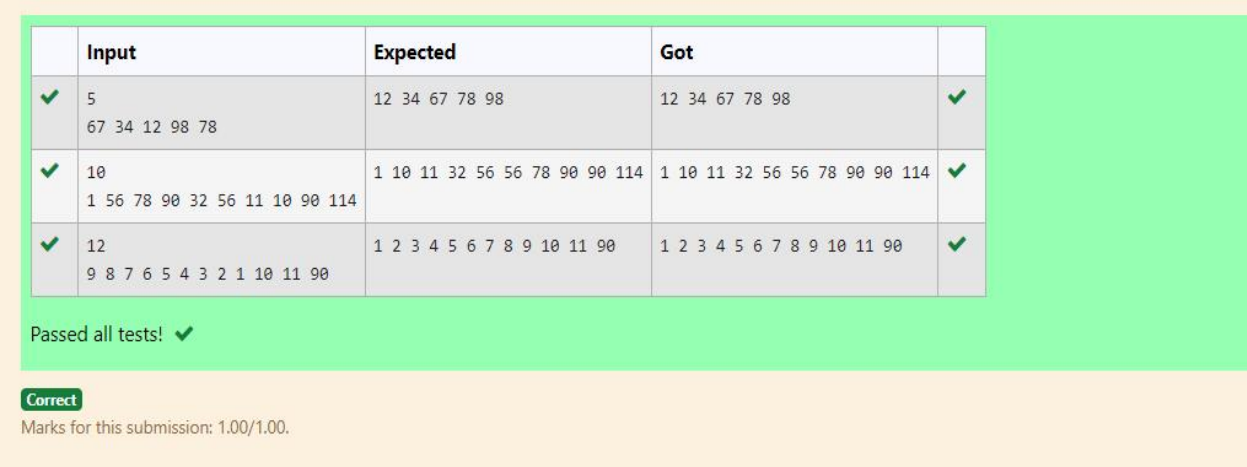
PROGRAM:

```
#include <stdio.h>

int partition(int a[],int l,int h){
    int piv=a[l];
    int i=l,j=h,t;
    while(i<j){
        while(a[i]<=piv && i<h){
            i++;
        }
        while(a[j]>piv && j>l){
            j--;
        }
        if(i<j){
```

```
            t=a[i];
            a[i]=a[j];
            a[j]=t;
        }
    }
    t=a[l];
    a[l]=a[j];
    a[j]=t;
    return j;
}
void Quicksort(int a[],int l,int h){
    if(l<h){
        int par=partition(a,l,h);
        Quicksort(a,l,par-1);
        Quicksort(a,par+1,h);
    }
}
int main(){
    int n;
    scanf("%d",&n);
```

```
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    Quicksort(a,0,n-1);
    for(int i=0;i<n;i++){
        printf("%d ",a[i]);
    }
}
```

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! ✔

Correct

Marks for this submission: 1.00/1.00.

RESULT :

Hence the above program has been executed successfully..