Ex. No: 4 **Date: 03.09.24**

Register No.: 230701360 **Name: Tejushree sanjeevikumar**

Divide and Conquer

4.a. Number of Zeros in a Given Array

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.

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:

```
void function(int m){
    set c = 0
    set b = m / 2
    read m from input

    create array a of size m

    for i = 0 to m - 1:
        read a[i] from input

    set c = 0
    for i = 0 to b - 1:
        if a[i] == 0:
```

```
            increment c by 1


    for j = b to m - 1:
        if a[j] == 0:
            increment c by 1


    print c
}
```

Program:

```
#include<stdio.h>
int main()
{
    int m,c=0;
    scanf("%d",&m);
    int b=m/2;
    int a[m];
    for(int i=0;i<m;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<b;i++)
    {
       if(a[i]==0)
       {
           c=c+1;
       }
    }
    for(int j=b;j<m;j++)
```

```
    {
        if(a[j]==0)
        {
            c=c+1;
        }
    }
    printf("%d",c);
}
```

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	✔

4.b. Majority Element

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
```

Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 10^4`
- `-2^31 <= nums[i] <= 2^31 - 1`

Algorithm:

void function(int n){

set c = 0

set a = 0

read n from input

create array nums of size n

for i = 0 to n - 1:

read nums[i] from input

set c = 0

for i = 0 to n - 1:

for j = 0 to n - 1:

if nums[i] == nums[j]:

```
            increment c by 1
        if c > c / 2:
            set a = nums[i]
            break

    print a
}
```
Program:

```
#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 c=0,a;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(nums[i]==nums[j])
            {
                c=c+1;
            }
            if(c>c/2)
            {
```

```
            a=nums[i];
            break;
        }

    }

}
printf("%d",a);
}
```

Output:

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

4.c. Finding Floor Value

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.

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:

```
void function(int n, int x){
    read n from input
    create array nums of size n

    for i = 0 to n - 1:
        read nums[i] from input

    read x from input

    set left = 0
    set right = n - 1
    set floor = -1

    while left <= right:
        set mid = (left + right) / 2

        if nums[mid] == x:
            set floor = nums[mid]
```

```
            break

        if nums[mid] < x:
            set floor = nums[mid]
            set left = mid + 1

        else:
            set right = mid - 1

    if floor == -1:
        print "No floor found"
    else:
        print floor
}
```

Program:

```
#include <stdio.h>

int main() {
    int n, x;



    scanf("%d", &n);

    int nums[n];



    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
```

```
}


scanf("%d", &x);


int left = 0;
int right = n - 1;
int floor = -1;
while (left <= right) {
    int mid = (left + right) / 2;

    if (nums[mid] == x) {
        floor = nums[mid];
        break;
    }

    if (nums[mid] < x) {
        floor = nums[mid];
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}


if (floor == -1) {
    printf("No floor found\n");
```

```
    } else {
        printf("%d\n", floor);
    }

    return 0;
}
```

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	✔

4.d. Two Elements Sum to X

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
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:

```
void function(int n, int x){
    set e1 = -1
    set e2 = -1
    read n from input
    create array nums of size n

    for i = 0 to n - 1:
        read nums[i] from input

    read x from input

    set left = 0
    set right = n - 1

    while left <= right:
        set mid = (left + right) / 2
```

```
        if nums[mid] + nums[right] == x:
            set e1 = nums[mid]
            set e2 = nums[right]
            break

        set right = right - 1

    if e1 > -1 and e2 > -1:
        print e1
        print e2
    else:
        print "No"
```
}**Program:**

```
#include <stdio.h>

int main() {
    int n, x,e1=-1,e2=-1;



    scanf("%d", &n);

    int nums[n];



    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
```

```
    }



    scanf("%d", &x);



    int left = 0;
    int right = n - 1;

    while (left <= right) {
        int mid = (left + right) / 2;

        if (nums[mid]+nums[right] == x) {
            e1 = nums[mid];
            e2=nums[right];
            break;
        }
        right--;
    }
    if(e1>-1&&e2>-1)
    {
        printf("%d\n",e1);
        printf("%d",e2);
    }
    else{
        printf("No");
    }
}
```

Output:

	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔
✔	5 2 4 6 8 10 100	No	No	✔

4.e. Implementation of Quick Sort

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

Algorithm:

```
void function(int n){
    read n from input
    create array a of size n

    for i = 0 to n - 1:
        read a[i] from input

    call quick(a, 0, n - 1)

    for i = 0 to n - 1:
        print a[i]
}

void quick(int a[], int l, int r){
    set temp = 0
    if l < r:
        set x = a[r]
        set i = l - 1

        for j = l to r - 1:
```

```
            if a[j] < x:
                increment i by 1
                set temp = a[i]
                set a[i] = a[j]
                set a[j] = temp

        set temp = a[i + 1]
        set a[i + 1] = a[r]
        set a[r] = temp

        set mid = i + 1
        call quick(a, l, mid - 1)
        call quick(a, mid + 1, r)
}
```

Program:

```
#include<stdio.h>
int main()
{
 int n;
 scanf("%d",&n);
 int a[n];
 for(int i=0;i<n;i++){
    scanf("%d",&a[i]);
 }

 void quick(int*,int,int);
  quick(a,0,n-1);
 for(int i=0;i<n;i++)
```

```
 {
    printf("%d ",a[i]);
 }
}
 void quick(int a[],int l,int r){
    int temp;
    if(l<r)
    {
       int x =a[r];
       int i=l-1;
       for(int j=l;j<r;j++){
          if(a[j]<x){
             i++;
             temp=a[i];
             a[i]=a[j];
             a[j]=temp;
          }
       }
       temp =a[i+1];
       a[i+1]=a[r];
       a[r]=temp;
       int mid=i+1;
       quick(a,l,mid-1);
       quick(a,mid+1,r);
    }
 }
```

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	✔