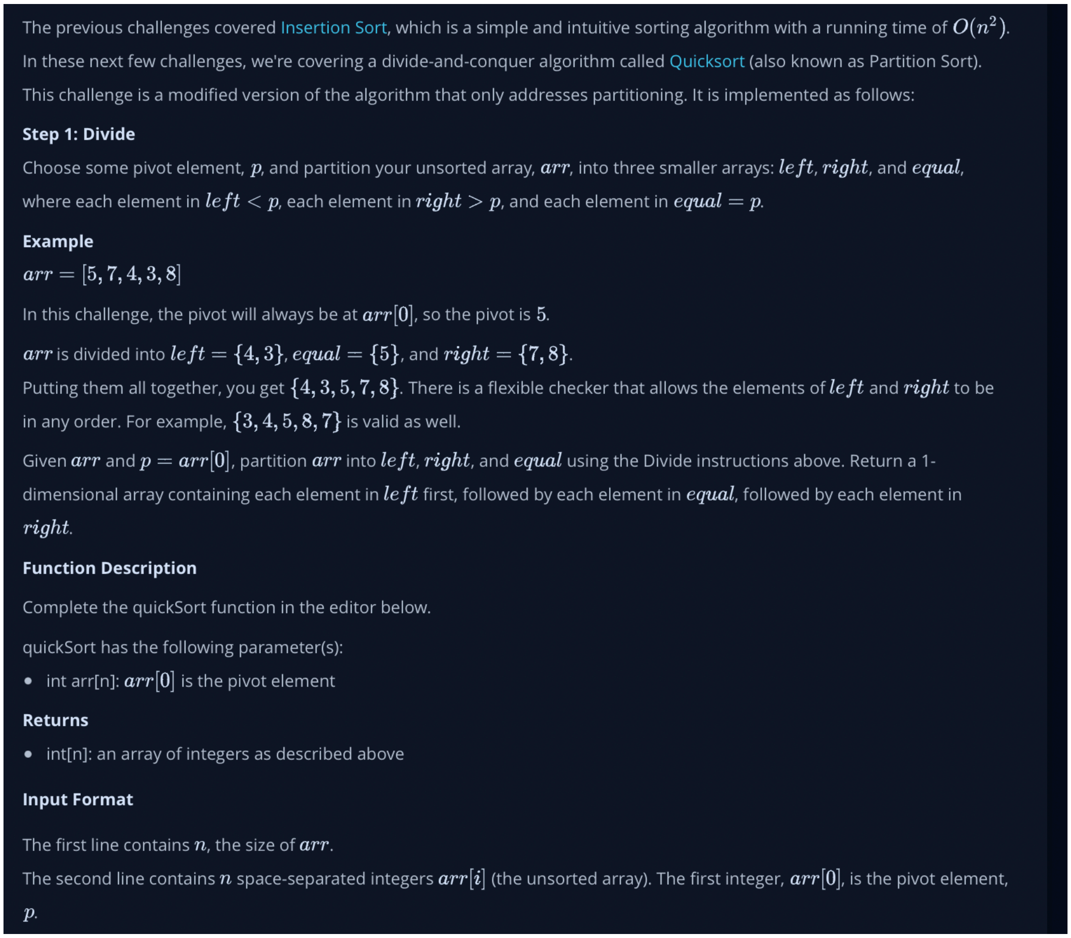
LabMst

Student Name:Rajiv Paul **UID:20BCS1812**
Branch: CSE **Section/Group: 20BCS_WM_702(A)**
Semester: 5th **Date of Performance:06/10/2022**
Subject Name: Competitive Coding **Subject Code: 20CSP-314**

1. Aim/Overview of the practical:



The previous challenges covered Insertion Sort, which is a simple and intuitive sorting algorithm with a running time of $O(n^2)$.

In these next few challenges, we're covering a divide-and-conquer algorithm called Quicksort (also known as Partition Sort).

This challenge is a modified version of the algorithm that only addresses partitioning. It is implemented as follows:

Step 1: Divide

Choose some pivot element, p, and partition your unsorted array, arr, into three smaller arrays: $left$, $right$, and $equal$, where each element in $left < p$, each element in $right > p$, and each element in $equal = p$.

Example

$arr = [5, 7, 4, 3, 8]$

In this challenge, the pivot will always be at $arr[0]$, so the pivot is 5.

arr is divided into $left = \{4, 3\}$, $equal = \{5\}$, and $right = \{7, 8\}$.

Putting them all together, you get $\{4, 3, 5, 7, 8\}$. There is a flexible checker that allows the elements of $left$ and $right$ to be in any order. For example, $\{3, 4, 5, 8, 7\}$ is valid as well.

Given arr and $p = arr[0]$, partition arr into $left$, $right$, and $equal$ using the Divide instructions above. Return a 1-dimensional array containing each element in $left$ first, followed by each element in $equal$, followed by each element in $right$.

Function Description

Complete the quickSort function in the editor below.

quickSort has the following parameter(s):

- int arr[n]: $arr[0]$ is the pivot element

Returns

- int[n]: an array of integers as described above

Input Format

The first line contains n, the size of arr.

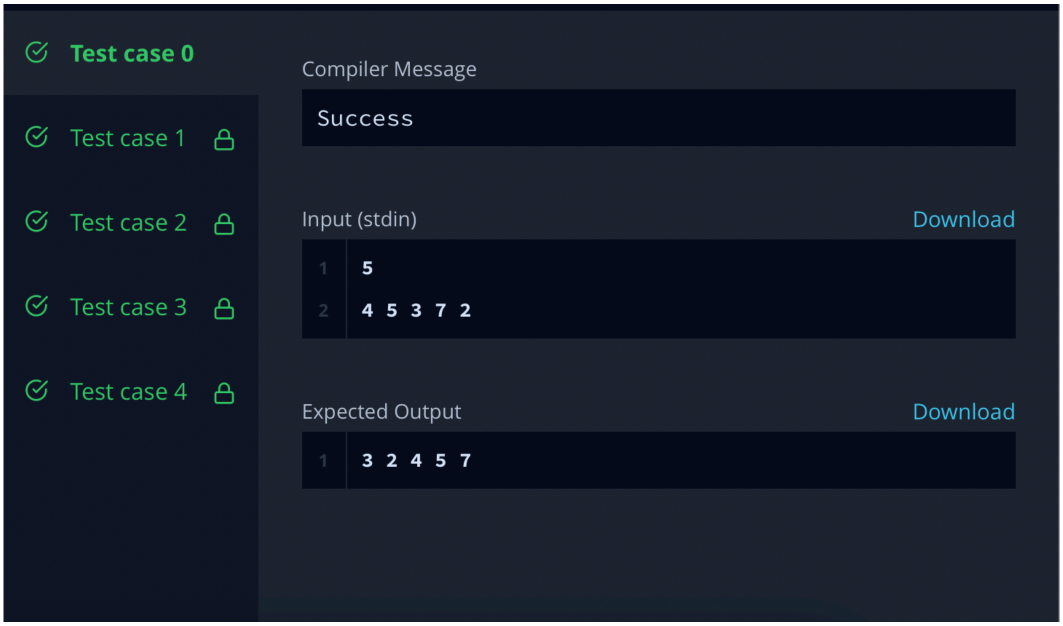
The second line contains n space-separated integers $arr[i]$ (the unsorted array). The first integer, $arr[0]$, is the pivot element, p.

3. Steps for experiment/practical/Code:

```
import java.util.*;

public class QuickSort_1_partition {
    public static void main(String[] args) {
        Scanner I = new Scanner(System.in);
        int n = I.nextInt();
        int []arr = new int[n];
        for (int i = 0; i < n ; i++) {
            arr[i]= I.nextInt();
        }
        partition(arr);
        printArr(arr);
    }
    static void printArr(int []arr){
        for(int n: arr){
            System.out.print(n+" ");
        }
        System.out.println("");
    }
    static void partition(int [] arr){
        int r = arr[0];
        int [] cp= Arrays.copyOf(arr,arr.length);
        int c =0;
        for (int i = 1; i < arr.length ; i++) {
            if(cp[i]<=r){
                arr[c]=cp[i];
                c++;
            }
        }
        arr[c]=r;
        c++;
        for (int i = 0; i < arr.length ; i++) {
            if (cp[i] > r) {
                arr[c]=cp[i];
                c++;
            }
        }
    }
}
```

4. Result/Output/Writing Summary:



Compiler Message

```
Success
```

Input (stdin)

```
5
4 5 3 7 2
```

Expected Output

```
3 2 4 5 7
```

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			