ADS Assignment-1

Problem 1:

Given an array of integers, perform the following operations:

1. Find the second largest element in the array.

2. Move all zeros to the end of the array while maintaining the order of non-zero elements.

Input:

arr = [10, 0, 5, 20, 0, 8, 15]

Output:

Second largest element: 15

Array after moving zeros: [10, 5, 20, 8, 15, 0, 0]

Constraints:

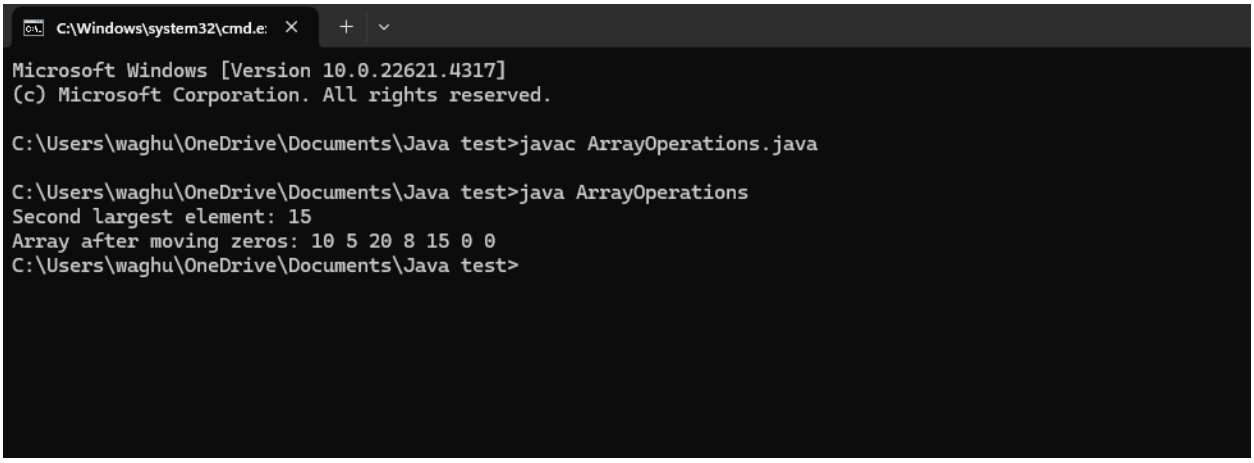
- Do not use built-in sort functions.
- The array may contain duplicate elements or zeros at any position.
- Array length ≥ 2.

ArrayOperations.java

```java
public class ArrayOperations {
    public static void main(String[] args) {
        int[] arr = {10, 0, 5, 20, 0, 8, 15};
        int secondLargest = findSecondLargest(arr);
        System.out.println("Second largest element: " + secondLargest);

        moveZerosToEnd(arr);
        System.out.print("Array after moving zeros: ");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
    public static int findSecondLargest(int[] arr) {
        int largest = Integer.MIN_VALUE, secondLargest = Integer.MIN_VALUE;
        for (int num : arr) {
            if (num > largest) {
                secondLargest = largest;
                largest = num;
            } else if (num > secondLargest && num != largest) {
                secondLargest = num;
            }
        }
        return secondLargest;
    }
    public static void moveZerosToEnd(int[] arr) {
        int index = 0;
        for (int num : arr) {
            if (num != 0) {
                arr[index++] = num;
            }
        }
        while (index < arr.length) {
            arr[index++] = 0;
        }
    }
}

```



Problem 2:

Write a program that performs the following operations on strings:

1. Check whether two given strings are anagrams of each other.

2. Identify the longest word in a given sentence.

3. Count the number of vowels and consonants in the same sentence.

Input:

String 1: listen

String 2: silent Sentence: Practice makes a man perfect

Output: Are 'listen' and 'silent' anagrams? true

Longest word: Practice

Vowels: 9, Consonants: 17

------ import java.util.Scanner;

```
public class Operations{
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		System.out.println("Input 2 Strings: ");
		String str1 = sc.nextLine();
		String str2 = sc.nextLine();
		System.out.println(str1 + " & " + str2 + " are anagrams? " + checkAnagram(str2, str1));
		System.out.println("Input a sentence: ");
		str1 = sc.nextLine();
		System.out.println("Longest Word in " + str1 + " is: " + findLargestWord(str1));
		int[] res = findVC(str1);
		System.out.println("Vowels: " + res[0] + ", Constants: " + res[1]);
	}

	public static String findLargestWord(String s){
		String[] words = s.split("\\s+");
		String longword = "";
		for(int i=0; i<words.length; i++){
			if(words[i].length()>longword.length())
				longword = words[i];
		}
		return longword;
	}
```

```
	public static boolean checkAnagram(String s1, String s2){
		if(s1.length()!=s2.length()) return false;
		s1 = s1.toLowerCase();
		s2 = s2.toUpperCase();
		int frequency[] = new int[26];
		for(int i=0; i<s1.length(); i++){
			frequency[s1.charAt(i)-'a']++;
			frequency[s2.charAt(i)-'a']--;
		}
		for(int i=0; i<frequency.length; i++){
			if(frequency[i]!=0) return false;
		}
		return true;
	}

	public static int[] findVC(String s){
		s = s.toLowerCase();
		String[] words = s.split("\\s+");
		int count1=0, count2=0;
		for(int i=0; i<words.length; i++){
			for(int j=0; j<words[i].length(); j++){
				if(words[i].charAt(j)=='a' || words[i].charAt(j)=='i' ||
words[i].charAt(j)=='e' || words[i].charAt(j)=='o' || words[i].charAt(j)=='u'){
					count1++;
				} else {
					count2++;
				}
			}
		}
```

```
            return new int[]{count1++, count2++};
        }
}
```

Problem 3:

Given a sorted array of integers (which may include duplicates), perform the following operations:

1. Search for a given key and return its index (if found) with Binary Search.
2. Find the first and last occurrence of the key in the array.
3. Count the total number of times the key appears.
4. Find any peak element in the array (an element greater than its neighbors).

Input:

arr = [1, 3, 3, 3, 5, 6, 8], key = 3

Input for Peak Element:

arr =[1, 2, 18, 4, 5, 0]

Output:

Key found at index: 2

First occurrence: 1

Last occurrence: 3

Total count of key: 3

Peak element: 18

```
import java.util.Arrays;
public class Array{
    public static void main(String[] args){
        int arr[] = {1,3,3,3,5,6,8};
        int ans = BinarySearch(arr, 3);
        if(ans!=-1){
            System.out.println("Key found at index: " + ans);
        } else {
            System.out.println("Element not found");
        }
```

```
        int[] res = findOccurances(arr, 3);
        System.out.println("First occurance: " + res[0] + "\nLast occurance: " + res[1]);
        System.out.println("Total count of key: " + countOccurances(arr, 3));
        int array[] = {1,2,18,4,5,0};
        System.out.println("Peak Element: " + peakElement(array));

}

public static int BinarySearch(int[] arr, int x){
        int low = 0, high = arr.length-1;
        while(low<=high){
                int mid = low + (high - low)/2;
                if(arr[mid]==x)
                        return mid;
                else if(arr[mid]<x)
                        high = mid-1;
                else
                        low = mid + 1;
        }
        return -1;
}

public static int[] findOccurances(int[] arr, int key){
        int firstIndex=-1, lastIndex=-1;
        for(int i=0; i<arr.length; i++){
                if(arr[i] == key){
                        if(firstIndex==-1)
                                firstIndex = i;
```

```
                    lastIndex=i;
                }
            }
            return new int[]{firstIndex, lastIndex};
        }

        public static int countOccurances(int[] arr, int key){
            int count=0;
            for(int i=0; i<arr.length; i++){
                if(arr[i] == key){
                    count++;
                }
            }
            return count;
        }

        public static int peakElement(int[] arr){
            int peak=arr[0];
            for(int i=1; i<arr.length; i++)
                peak = arr[i]>peak?arr[i]: peak;
            return peak;
        }
}
```

Problem 4:

Write a recursive program that performs the following operations:

1. Check if a number is prime using recursion.

2. Check whether a given string is a palindrome.

3. Find the sum of digits of a given number.

4. Calculate the nth Fibonacci number.

```
----  public class Problem4{
        public static void main(String[] args){
                int x=27;
                System.out.println("Is prime: " + findPrime(x, 2));
                String s = "racecar";
                System.out.println("Is '" + s + "' a palindrome? " + checkPalindrome(s,0,s.length()-1));
                x=1234;
                System.out.println("Sum of digits of " + x + ": " + digitSum(x));
                x=6;
                System.out.println("Fibonacci(" + x + "): " + fibonacci(0,1,x,0));
                x=2;
                int y=5;
                System.out.println(x + "^" + y + " = " + power(x,y));
        }

        public static boolean checkPalindrome(String s, int low, int high){
                if(low>=high){
                        return true;
                } else {
                        if(s.charAt(low)!=s.charAt(high))
                                return false;
                        return checkPalindrome(s,low+1,high-1);
                }
        }

        public static int digitSum(int n){
                if(n==0){
                        return 0;
```

```
        } else {
                return (n%10)+digitSum(n/10);
        }
}

public static boolean findPrime(int x,int i){
        if(i>=(int)Math.sqrt(x)){
                return true;
        } else {
                if(x%i==0)
                        return false;
                return findPrime(x,i+1);
        }
}

public static int fibonacci(int x, int y, int n, int i){
        if(i==n){
                return x;
        } else {
                return fibonacci(y, x+y, n, i+1);
        }
}

public static int power(int x, int y){
        if(y==1){
                return x;
        } else {
                return x*power(x, y-1);
        }
```

```
        }
}
```

5. Calculate a raised to the power b

Input:

num = 7

str = "racecar"

num = 1234

fibIndex = 6

a = 2, b = 5

Output:

Is prime: true

Is 'racecar' a palindrome? true

Sum of digits of 1234: 10

Fibonacci(6): 8

2^5 = 32

Constraints:

- Do not use loops or built-in reverse methods.
- Use charAt() for string access.
- You can assume valid positive integer inputs.

```
----- public class RecursiveOperations {
    public static boolean isPrime(int num, int divisor) {
    if (num < 2) return false;
    if (divisor == 1) return true;
    if (num % divisor == 0) return false;
    return isPrime(num, divisor - 1);
  }
    public static boolean isPalindrome(String str, int left, int right) {
    if (left >= right) return true;
    if (str.charAt(left) != str.charAt(right)) return false;
```

```
    return isPalindrome(str, left + 1, right - 1);
}

public static int sumOfDigits(int num) {
    if (num == 0) return 0;
    return (num % 10) + sumOfDigits(num / 10);
}

public static int fibonacci(int n) {
    if (n <= 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

public static int power(int a, int b) {
    if (b == 0) return 1;
    return a * power(a, b - 1);
}

public static void main(String[] args) {
    int num = 7;
    System.out.println("Is prime: " + isPrime(num, num - 1));

    String str = "racecar";
    System.out.println("Is '" + str + "' a palindrome? " + isPalindrome(str, 0, str.length() - 1));

    int digitSum = 1234;
    System.out.println("Sum of digits of " + digitSum + ": " + sumOfDigits(digitSum));
```

```
        int fibIndex = 6;
        System.out.println("Fibonacci(" + fibIndex + "): " + fibonacci(fibIndex));

        int a = 2, b = 5;
        System.out.println(a + "^" + b + " = " + power(a, b));
    }
}
```