Question 1:

Given a string "s", reverse only all the vowels in the string and return it.
The vowels are 'a', 'e', 'ï', 'o', and 'u', and they can appear in both lower and upper
cases, more than once.

Example 1:
Input: s = "hello"
Output: “holle”

Example 2:
Input: s = "zoho corporation"
Output: "zohi carporotoon”

```java
public class Q1 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		String input = "Hello";
		int start = 0;
		int end = input.length() - 1;

		char[] chararray = input.toCharArray();
		changeVowelPositions(chararray, start, end);

		String result = new String(chararray);
		System.out.println(result);

	}

	static void changeVowelPositions(char[] chararray, int start, int end) {
		while (start < end) {

			while (start < end && !isVowel(chararray[start])) {
				start++;
			}

			while (start < end && !isVowel(chararray[end])) {
				end--;
			}

			if (start < end) {
				swap(chararray, start, end);
				start++;
				end--;

			}

		}
	}

	static void swap(char[] chararray, int start, int end) {
		char temp = chararray[start];
		chararray[start] = chararray[end];
		chararray[end] = temp;
	}

	static boolean isVowel(char c) {
		return "aeiouAEIOU".indexOf(c) != -1;
	}

}
```

Generate Parenthesis:

Given an integer n, representing the number of pairs of parentheses, write a function to generate all possible combinations of well-formed parentheses.
Example 1:
Input: n = 3
Output: [“((()))”,”(()())”,”(())()”:”()(())”, “()()()”]

Explanation: For n = 3, the function should generate all possible combinations of well-formed parentheses.
The valid combinations are: [“((()))”,”(()())”,”(())()”:”()(())”, “()()()”]
Example 2:
Input: n = 1
Output: [“()”]
Explanation: For n = 1, the function should generate all possible combinations of well-formed parentheses.
The valid combination is only "()".
Constraints: 1 <=n<= 8
Note: The order of the output does not matter.
Each pair of parentheses should be well-formed, meaning that for every open parenthesis there is a corresponding closing parenthesis, and the parentheses are properly nested.

Solution:

```java
// Print num of valid paranthasis
public class Q2 {
    static int count = 0;

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int n = 2;
        int open = 0;
        int close = 0;
        validParanthasis("", open, close, n);
        System.out.println(count);

    }

    static void validParanthasis(String s, int open, int
close, int n) {
        if (s.length() == 2 * n) {
            count++;
            System.out.println(s);
            return;
        }
        if (open < n) {
            validParanthasis(s + "(", open + 1, close, n);
        }
        if (close < open) {
            validParanthasis(s + ")", open, close + 1, n);
        }
    }

}
```

3. Given a pattern and a string "s", find if "s" follows the same pattern.
Here follow means a full match, such that there is a bisection between a letter in
pattern and a non-empty word in s.
Example 1:
Input: pattern = "abba", s = "dog cat cat dog" Output: true
Example 2:
Input: pattern = "abba", s = "dog cat cat fish" Output: false
Example 3:
Input: pattern = "aaaa", s = "dog cat cat dog" Output: false

```java
import java.util.HashMap;

public class Q3 {

	public static void main(String[] args) {

		String first = "abba";
		String second = "dog cat cat dog";

		System.out.println(checkfollows(first, second));
	}

	static boolean checkfollows(String first, String Second) {
		String[] words = Second.split(" ");
		if (first.length() != words.length) {
			return false;
		}
		HashMap<Character, String> map = new HashMap<>();

		for (int i = 0; i < first.length(); i++) {

			if (!map.containsKey(first.charAt(i))) {

				map.put(first.charAt(i), words[i]);

			} else {
				String val = map.get(first.charAt(i));
				if (!val.equals(words[i])) {
					return false;
				}

			}

		}

		return true;
	}

}
```

4.

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the

median of the two sorted arrays.

Example 1:

Input: nums1 = [1,3], nums2 = [2]

Output: 2.00000

Explanation: merged array = [1,2,3] and median is 2.

Example 2:

Input: nums1 = [1,2], nums2 = [3,4]

Output: 2.50000

Explanation: merged array = [1,2,3,4] and median is (2+3)/2=2.5.

```java
import java.util.Arrays;

// Median a=of two sorted arrays when merge into one

public class Q4{

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		int[] first = { 1 ,3};
		int[] second = { 2 };

		int i = first.length;
		int j = second.length;

		int[] mergearray = new int[i + j];

		i = 0;
		j = 0;
		int k = 0;

		while (i < first.length && j < second.length) {
			if (first[i] < second[j]) {
				mergearray[k++] = first[i++];
			} else {
				mergearray[k++] = second[j++];
			}

		}

		while (i < first.length) {
			mergearray[k++] = first[i++];
		}

		while (j < second.length) {
			mergearray[k++] = second[j++];
		}

		System.out.println(Arrays.toString(mergearray));

		if (mergearray.length % 2 == 1) {
			System.out.println(mergearray[mergearray.length / 2]);
		} else {
			System.out.println(
					(double) (mergearray[mergearray.length / 2 - 1] +
mergearray[mergearray.length / 2 ]) / 2);

		}

	}

}
```

Question 5:

You are given an n x n 2D matrix representing an image Your task is to rotate the image by 90 degrees clockwise. The rotation should be done in place, meaning you have to modify the input 2D matrix directly. DO NOT allocate another 2D matriz

Input: A square 2D matrix of size n x n, where n is the number of rows and columns

Output: The input matrix rotated by 90 degrees clockwise.

Example 1:

Input Matrix: [
[1, 2, 3],
[4, 5, 6].
[7,8,9]
]

Output Matrix: [
[7.4, 1],
[8, 5, 2],
[9,6,3]]

Example 2:

Input Matrix: [
[5, 1, 9, 11],
[2, 4, 8, 10],
[13, 3, 6, 7],
[15, 14, 12, 16]
]

Output Matrix: [
[15, 13, 2, 5],
[14, 3, 4, 1],
[12, 6, 8, 9],
[16, 7, 10, 11]]

```java
// Rotate matrix by 90 degrees
public class Q5 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		int[][] arr = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };


		int count = 0;

		// Tranpose
		for (int i = 0; i < arr.length; i++) {
			for (int j = i + 1; j < arr.length; j++) {
				System.out.println(" Count : " + count);
				int temp = arr[i][j];
				arr[i][j] = arr[j][i];
				arr[j][i] = temp;

			}
		}

		// revrese the row
		for (int i = 0; i < arr.length; i++) {
			for (int j = 0; j < arr.length / 2; j++) {
				int temp = arr[i][arr.length - 1 - j];
				arr[i][arr.length - 1 - j] = arr[i][j];
				arr[i][j] = temp;

			}
		}

		// print
		for (int i = 0; i < arr.length; i++) {
			for (int j = 0; j < arr.length; j++) {
				System.out.print(arr[i][j] + " ");

			}
			System.out.println();
		}

	}

}
```