Assignment 4 Solution - 2D Arrays | DSA

Question 1

Given three integer arrays arr1, arr2 and arr3 sorted in strictly increasing order, return a sorted array of only the integers that appeared in all three arrays.
Example 1:
Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]
Output: **[1,5]**
Explanation: Only 1 and 5 appeared in the three arrays.

Solution Code:

```
package in.ineuron.pptAssignment04;
import java.util.ArrayList;
import java.util.List;
public class CommonElements {
    public static int[] findCommonElements(int[] arr1, int[] arr2, int[] arr3) {
        int i = 0, j = 0, k = 0;
        List<Integer> result = new ArrayList<>();

        while (i < arr1.length && j < arr2.length && k < arr3.length) {
            if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {
                result.add(arr1[i]);
                i++;
                j++;
                k++;
            } else if (arr1[i] < arr2[j]) {
                i++;
            } else if (arr2[j] < arr3[k]) {
                j++;
            } else {
                k++;
            }
        }

        int[] output = new int[result.size()];
        for (int m = 0; m < result.size(); m++) {
            output[m] = result.get(m);
        }

        return output;
    }

    public static void main(String[] args) {
        int[] arr1 = { 1, 2, 3, 4, 5 };
        int[] arr2 = { 1, 2, 5, 7, 9 };
        int[] arr3 = { 1, 3, 4, 5, 8 };
```

devavrat.wadekar@gmail.com

```
                int[] result = findCommonElements(arr1, arr2, arr3);

                System.out.print("Output: [");
                for (int i = 0; i < result.length; i++) {
                        if (i != 0) {
                                System.out.print(", ");
                        }
                        System.out.print(result[i]);
                }
                System.out.println("]");
        }
}
```

Question 2

Given two 0-indexed integer arrays nums1 and nums2, return a list answer of size 2 where:
- answer[0] is a list of all distinct integers in nums1 which are not present in nums2 .
- answer[1] is a list of all distinct integers in nums2 which are not present in nums1.

Note that the integers in the lists may be returned in any order.

Example 1:

Input: nums1 = [1,2,3], nums2 = [2,4,6]

Output**: [[1,3],[4,6]]**

Explanation:

For nums1, nums1[1] = 2 is present at index 0 of nums2, whereas nums1[0] = 1 and nums1[2] = 3 are not present in nums2. Therefore, **answer[0] = [1,3].**

For nums2, nums2[0] = 2 is present at index 1 of nums1, whereas nums2[1] = 4 and nums2[2] = 6 are not present in nums2. Therefore, **answer[1] = [4,6].**

Solution Code:

```
package in.ineuron.pptAssignment04;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
public class FindDisjointArrays {
        public static int[][] findDisjointArrays(int[] nums1, int[] nums2) {
                Set<Integer> set1 = new HashSet<>();
                Set<Integer> set2 = new HashSet<>();

                for (int num : nums1) {
                        set1.add(num);
                }

                for (int num : nums2) {
                        set2.add(num);
                }
                List<Integer> list1 = new ArrayList<>();
                List<Integer> list2 = new ArrayList<>();
```

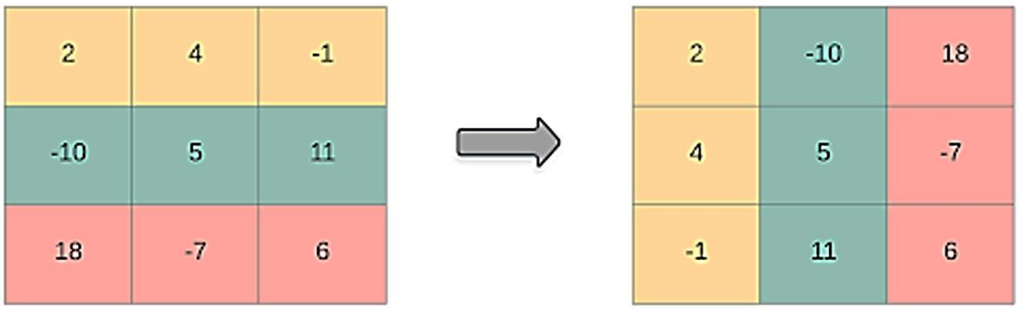
devavrat.wadekar@gmail.com

```
        for (int num : nums1) {
            if (!set2.contains(num)) {
                list1.add(num);
            }
        }

        for (int num : nums2) {
            if (!set1.contains(num)) {
                list2.add(num);
            }
        }

        int[][] answer = new int[2][];
        answer[0] = new int[list1.size()];
        answer[1] = new int[list2.size()];

        for (int i = 0; i < list1.size(); i++) {
            answer[0][i] = list1.get(i);
        }

        for (int i = 0; i < list2.size(); i++) {
            answer[1][i] = list2.get(i);
        }

        return answer;
    }
    public static void main(String[] args) {
        int[] nums1 = { 1, 2, 3 };
        int[] nums2 = { 2, 4, 6 };

        int[][] answer = findDisjointArrays(nums1, nums2);

        System.out.println("Distinct integers in nums1 not present in nums2: ");
        for (int num : answer[0]) {
            System.out.print(num + " ");
        }
        System.out.println();

        System.out.println("Distinct integers in nums2 not present in nums1: ");
        for (int num : answer[1]) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

devavrat.wadekar@gmail.com

Question 3

Given a 2D integer array matrix, return the transpose of matrix.
The transpose of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.
Example 1:
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: **[[1,4,7],[2,5,8],[3,6,9]]**



Solution Code :

```
package in.ineuron.pptAssignment04;
public class MatrixTranspose {
    public static int[][] transpose(int[][] matrix) {
        int rows = matrix.length;
        int columns = matrix[0].length;

        int[][] transposedMatrix = new int[columns][rows];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                transposedMatrix[j][i] = matrix[i][j];
            }
        }
        return transposedMatrix;
    }

    public static void main(String[] args) {
        int[][] matrix = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
        int[][] transposedMatrix = transpose(matrix);

        // Print the transposed matrix
        for (int i = 0; i < transposedMatrix.length; i++) {
            for (int j = 0; j < transposedMatrix[0].length; j++) {
                System.out.print(transposedMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

devavrat.wadekar@gm....com

Question 4

Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn) such that the sum of min(ai, bi) for all i is maximized . Return the maximized sum .

Example 1:

Input: nums = [1,4,3,2]

Output**: 4**

Explanation: All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4

So **the maximum possible sum is 4.**

Solution Code:

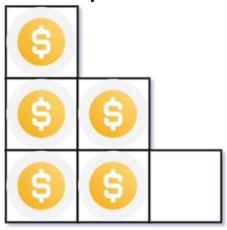
```
package in.ineuron.pptAssignment04;
import java.util.Arrays;

public class ArrayPairSum {

    public static int arrayPairSum(int[] nums) {
        // Sort the array in ascending order
        Arrays.sort(nums);

        int sum = 0;
        // Take the minimum element from each pair
        for (int i = 0; i < nums.length; i += 2) {
            sum += nums[i];
        }

        return sum;
    }

    public static void main(String[] args) {
        int[] nums = { 1, 4, 3, 2 };
        int maxSum = arrayPairSum(nums);
        System.out.println("Maximized sum: " + maxSum);
    }
}
```

devavrat.wadekar@gmail.com

Question 5

You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase may be incomplete. Given the integer n, return the number of complete rows of the staircase you will build .

Example 1:



Input: n = 5

Output: **2**

Explanation: Because the 3rd row is incomplete, we return 2.

Solution Code:

```
package in.ineuron.pptAssignment04;

public class Staircase {
        public static int countCompleteRows(int n) {
                int row = 1;
                while (n >= row) {
                        n -= row;
                        row++;
                }
                return row - 1;
        }

        public static void main(String[] args) {
                int n = 5;
                int completeRows = countCompleteRows(n);
                System.out.println("Number of complete rows: " + completeRows);
        }
}
```

devavrat.wadekar@gmail.com

Question 6

Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order .

Example 1:

Input: nums = [-4,-1,0,3,10]

Output**: [0,1,9,16,100]**

Explanation: After squaring, the array becomes [16,1,0,9,100].

After sorting, it becomes [0,1,9,16,100]

Solution Code:

```
package in.ineuron.pptAssignment04;
import java.util.Arrays;

public class SortedSquares {
    public int[] sortedSquares(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];
        int left = 0;
        int right = n - 1;
        int index = n - 1;

        while (left <= right) {
            int leftSquare = nums[left] * nums[left];
            int rightSquare = nums[right] * nums[right];

            if (leftSquare > rightSquare) {
                result[index] = leftSquare;
                left++;
            } else {
                result[index] = rightSquare;
                right--;
            }

            index--;
        }

        return result;
    }

    public static void main(String[] args) {
        int[] nums = { -4, -1, 0, 3, 10 };
        SortedSquares obj = new SortedSquares();
        int[] result = obj.sortedSquares(nums);
        System.out.println(Arrays.toString(result));
    }
}
```

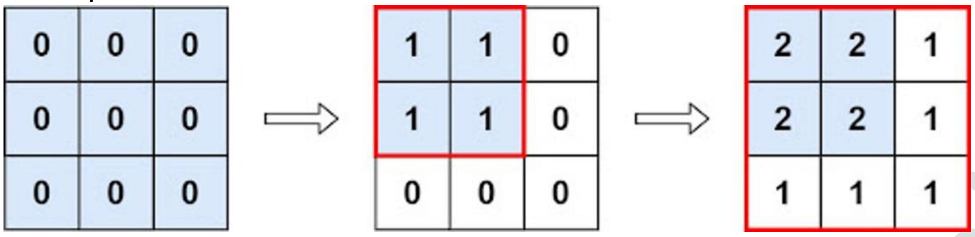
devavrat.wadekar@gmail.com

Question 7

You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.

Count and return the number of maximum integers in the matrix after performing all the operations

Example 1:



Input: m = 3, n = 3, ops = [[2,2],[3,3]]

Output: **4**

Explanation: The maximum integer in M is 2, and there are four of it in M. So return 4.

Solution Code:

```
package in.ineuron.pptAssignment04;

public class MatrixOperations {
    public static void main(String[] args) {
        int[][] ops = { { 2, 2 }, { 3, 3 } };
        int m = 3, n = 3;
        System.out.println("Matrix Operations :: "+maxCount(m, n, ops));
    }

    public static int maxCount(int m, int n, int[][] ops) {
        if (ops == null || ops.length == 0) {
            return m * n;
        }

        int minX = Integer.MAX_VALUE;
        int minY = Integer.MAX_VALUE;

        for (int[] op : ops) {
            minX = Math.min(minX, op[0]);
            minY = Math.min(minY, op[1]);
        }

        return minX * minY;
    }
}
```

devavrat.wadekar@gmail.com

Question 8

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].
Return the array in the form [x1,y1,x2,y2,...,xn,yn].
Example 1:
Input: nums = [2,5,1,3,4,7], n = 3
Output**: [2,3,5,4,1,7]**
Explanation: Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the **answer is [2,3,5,4,1,7].**

Solution Code:

```
package in.ineuron.pptAssignment04;

public class ShuffleArray {
	public static int[] shuffle(int[] nums, int n) {
		int[] result = new int[2 * n];
		int index = 0;

		// Traverse through the array elements
		for (int i = 0; i < n; i++) {
			// Add x[i] and y[i] to the result array
			result[index++] = nums[i];
			result[index++] = nums[i + n];
		}

		return result;
	}

	public static void main(String[] args) {
		int[] nums = { 2, 5, 1, 3, 4, 7 };
		int n = 3;

		int[] shuffledArray = shuffle(nums, n);

		// Print the shuffled array
		for (int num : shuffledArray) {
			System.out.print(num + " ");
		}
	}
}
```

devavrat.wadekar@gmail.com