EECS1022 – Lab 4: Arrays

Fall 2025 **Due Date: Sunday, October 26(11:59pm)**

Overview

Welcome to Lab 4!

This graded lab assignment focuses on advanced array processing, two-dimensional array operations, and string manipulation techniques. Students will implement five utility methods that demonstrate proficiency with one-dimensional and two-dimensional arrays, nested loop structures, and complex pattern recognition algorithms.

Lab Policies

- **Academic Integrity**: Submit your own work. Do not copy code from classmates or online sources. All violations will be reported as academic misconduct.
- **Submission Format**: Submit only the file: **Lab4.java**. Do not upload ZIP files or project folders. Do not change the name of the file as this will not allow us to test your code.
- **Deadline:** Submit your .java file to the eClass course page by **Sunday, October26 (11:59pm).** No late submissions are accepted. Email submissions are not accepted.
- Your lab assignment is not graded during the weekly lab sessions scheduled.

Learning Outcomes

By the end of this lab, you will be able to:

- Import a starter project archive file.
- Process one-dimensional and two-dimensional arrays with efficient traversal techniques
- Perform pattern recognition and sequence analysis in integer arrays
- Generate formatted string outputs with specific formatting requirements
- Implement case-insensitive string subsequence matching algorithms
- Handle edge cases including empty arrays, single elements, and boundary conditions
- Apply nested loop concepts for two-dimensional array problems
- Use the given JUnit tests (calling the utility methods) to guide the development.

EECS1022 – Lab 4: Arrays

Fall 2025 **Due Date: Sunday, October 26(11:59pm)**

Lab Requirements

- For the JUnit test class Lab4Test.java given to you:
  - Do not modify the test methods given to you.
  - You are allowed to add new test methods.
- For each method in the Lab4 class that you are assigned to implement:
  - Javadoc has been generated for you. You can check the documentation of the project in your browser by clicking on index.html file.
  - No System.out.println statements should appear in each of the utility method.
  - No Scanner operations (e.g., input.nextInt()) should appear in each of the utility methods. Instead, refer to the input parameters of the method.
  - No main method.
  - NO RECURSION - No recursive method calls
  - Don't change the method signatures provided to you. Otherwise, your test cases will fail.
  - Complete the partial code in Lab4.java file to achieve desired solution for the listed programming tasks.
  - A tester file Lab4Test.java is provided to you with some of the test cases implemented.

Download and Import the Starter Project

- Download the Eclipse Java project archive file from eClass: EECS1022_Lab4.zip
- Launch Eclipse and browse to EECS1022-workspace (for instance or your own created workspace).
- In Eclipse:
  - Choose File->Import
  - Under General, choose Existing Projects into workspace
  - Choose Select archive file. Browse your compressed zip folder and attach it.
  - Make sure that the EECS1022_Lab4 box is checked under Projects and you don't have the same project already in the workspace. Then Finish.

EECS1022 – Lab 4: Arrays

Fall 2025 **Due Date: Sunday, October 26(11:59pm)**

Programming Tasks

You are required to implement the following three methods in the Lab4.java class

Method 1: longestDecreasingSeq()

Write a method named longestDecreasingSeq that takes as a parameter an array of integers. The method should return the length of the longest consecutive sequence of decreasing integers in the array.

Method Signature: **public static int longestDecreasingSeq(int[] arr)**

Examples:

Call	Value Returned
longestDecreasingSeq({15,10,9,8,5,12,7,6})	5
longestDecreasingSeq({1,5,9,12,15})	1
longestDecreasingSeq({})	0

Method 2: frequencyArray()

Write a method frequencyArray that takes an array of integers as a parameter. The method checks for odd integers in range 1 inclusive to 50 exclusive and counts how many occurrences each are in the array. The method should return a string that has all the qualifying values along with the number of their occurrences.

Method Signature: **public static String frequencyArray(int[] arr)**

Examples:

Call	Expected Output
frequencyArray({3, 15, 15, 49, 8, 721})	"3: 1 15: 2 49: 1”
frequencyArray({7,11,7,7,23,40})	"7: 3 11: 1 23: 1"
frequencyArray({})	“”

Method 3: canFormString()

Write a method named canFormString that takes 2 parameters, a string source and a string target. The method should determine if the characters in target appear in source in the same order (but not necessarily consecutively). The method should consider all strings irrespective of their case.

Method Signature: public static boolean canFormString(String source, String target)

Examples:

Call	Value Returned
canFormString("PROGRAMMING", "gain")	true
canFormString("alGorithm", "ago")	true
canFormString("computer", "pure")	false

Method 4: matrixOperation()

Write a method matrixOperation that takes an n-by-n array of integers. The method inspects every element in the array. It should calculate the sum of elements on the main diagonal (where row index equals column index) and the product of elements on the anti-diagonal (where row index + column index = n-1). The result should be returned as a string.

Method Signature: public static String matrixOperation(int[][] matrix)

Examples:

Call	Value Returned
matrixOperation({{2,4,6},{8,10,12},{14,16,18}})	"Diagonal Sum:30, Anti-Diagonal Product:840"
matrixOperation({})	""
matrixOperation({{5}})	"Diagonal Sum:5, Anti-Diagonal Product:5"

EECS1022 – Lab 4: Arrays

Fall 2025 **Due Date: Sunday, October 26(11:59pm)**

Method 5: latinSquare()

A Latin square of order n is an n-by-n array filled with n different symbols (typically the integers 1 to n) such that each symbol occurs exactly once in each row and exactly once in each column. Write a method called latinSquare that takes an n-by-n array and returns true if it forms a valid Latin square containing integers from 1 to n, false otherwise.

Method Signature: public static boolean latinSquare(int[][] matrix)

Examples:

Call	Value Returned
latinSquare({{1,2,3},{2,3,1},{3,1,2}})	true
latinSquare({{1,2,3},{1,3,2},{3,1,2}})	false
latinSquare({{1}})	true

Submission Instructions

- Submit only one file: **Lab4.java**
- Upload it to the Lab 4 assignment on eClass.
- Ensure your file compiles and runs without error.
- Submitting files with compilation errors will receive zero marks. No partial marks.
- Your file will be tested with additional test cases.