Lab 10 Computer Science 1027B

Introduction

The purpose of these labs is to provide practice problems for you to get used to the concepts being covered in class. It is very important that you do these labs on your own without relying on available AI tools. Questions on exams are often very similar to exercises presented in these labs, so understanding them and solving these problems gives you a huge advantage for the exams. Make sure that you complete both the code as well as provide written answers to the blue questions within the boxes.
When you have concluded this lab, you should submit the following files to OWL Brightspace:

- MatrixOperations.java
- UnderageException.java
- AgeVerification.java
- SinglyLinkedList.java
- quickSortVisual.<png/jpg>
- E05.txt
- written_answers.txt

Topics Covered

- Everything from weeks 1-6 (this is final exam practice)

Exercise 01 (Week 01)

Write a method, **multiplyMatrices**, that takes two 2D arrays '**matrixA**' and '**matrixB**' as input and returns their product as a new 2D array.

Ensure that the number of columns in '**matrixA**' matches the number of rows in '**matrixB**'. If not, the method should return 'null'.

Use a triple-nested for loop to computer the product of the two matrices.

```
public class MatrixOperations {
    public static int[][] multiplyMatrices(int[][] matrixA, int[][] matrixB) {
        // your code here
    }
}
```

Lab 10 Computer Science 1027B

Exercise 02 (Week 02)

You are tasked with writing a class, '**AgeVerification**', that checks if a person is old enough to access certain services. You will create a custom exception '**UnderageException**' to handle cases where the person does not meet the age requirement.

1. Create a custom exception class 'UnderageException':
    a. This exception should extend 'Exception' and have two constructors:
        i. A default constructor.
        ii. A constructor that accepts a custom error message.
2. Implement the 'AgeVerification' class:
    a. Fields:
        i. private String name.
        ii. private int age.
    b. Methods:
        i. public AgeVerification(String name, int age): Constructor to initialize the name and age.
        ii. public void verifyAge(int requiredAge) throw UnderageException: Method to check if the person's age is greater than or equal to the required age. This method should throw the custom 'UnderageException' when the age requirement is not met.
3. Demonstrate the usage of 'AgeVerification' and 'UnderageException' in a main method (within the AgeVerification class):
    a. Create an 'AgeVerification' instance.
    b. Verify the age for different age requirements.
    c. Catch and handle the 'UnderageException' when the age requirement is not met.

Q 02.1

Is UnderageException a checked or unchecked exception? Why?

Lab 10 Computer Science 1027B

Exercise 03 (Week 03)

Implement a singly-linked list by writing the class **SinglyLinkedList** given the Node class below.

```
public class Node {
    int data;
    Node next;

    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}
```

Then, write four methods:

addLast(int data): Adds a new node with the specified data at the end of the list.
addFirst(int data): Adds a new node with the specified data at the beginning of the list.
shiftRight(): Shifts every node over by one position to the right, moving the tail to the front.

- If the list is empty or has one element, no shift is needed

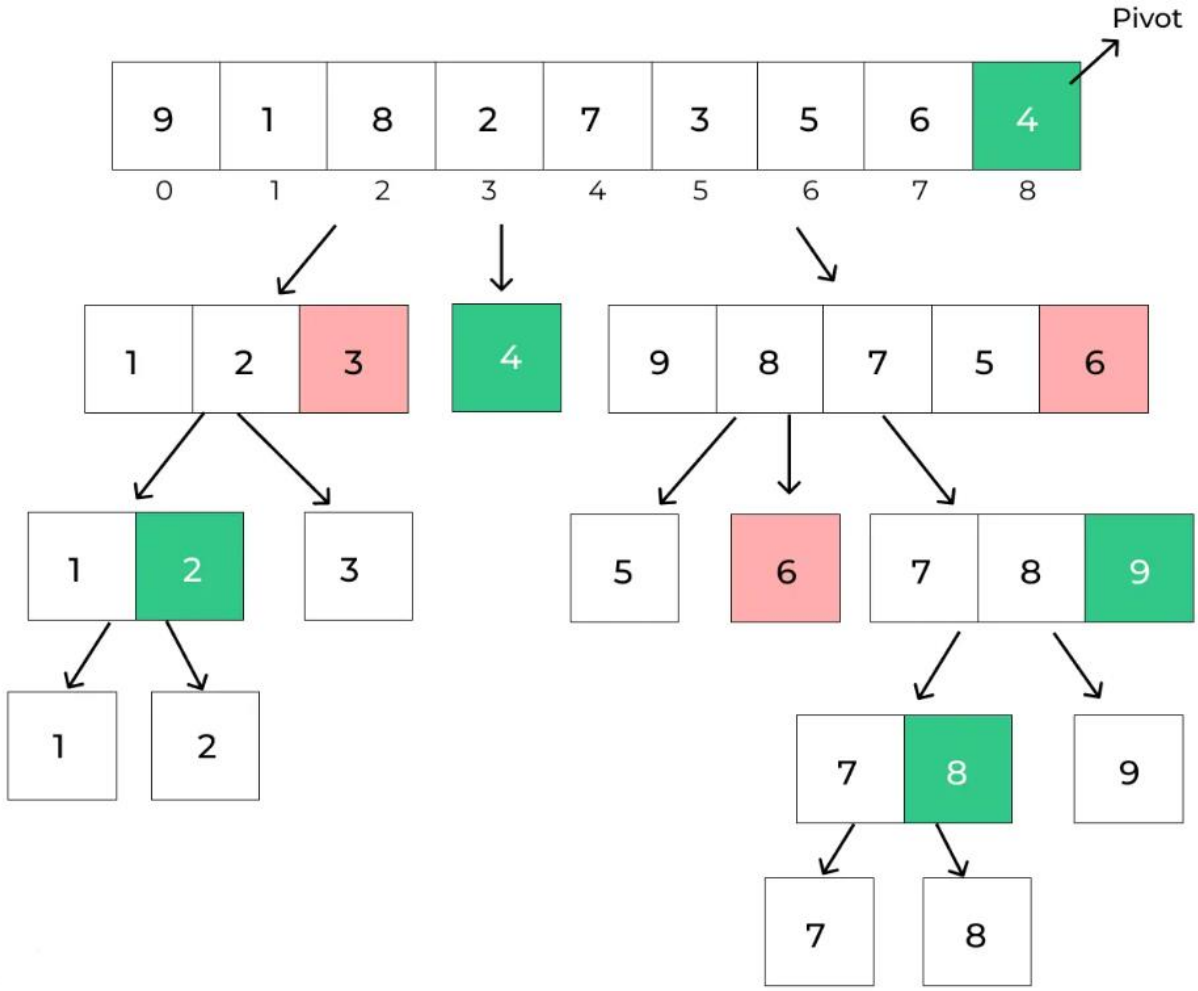
shiftLeft(): Shifts every node over by one position to the left, moving the head to the end.

- If the list is empty or has one element, no shift is needed.

Lab 10 Computer Science 1027B

Exercise 04 (Week 04)

If we were to sort an array containing the values **9, 1, 8, 2, 7, 3, 5, 6, 4** using quick sort, we can represent the entire process visually as:



Assume that you instead want to sort an array containing **-1, 3, 9, -10, 102, -4, -411, 512, 10**. Show a similar visual. You can draw this on paper or draw it digitally, as long as it's contained in a file called '**quickSortVisual**'.

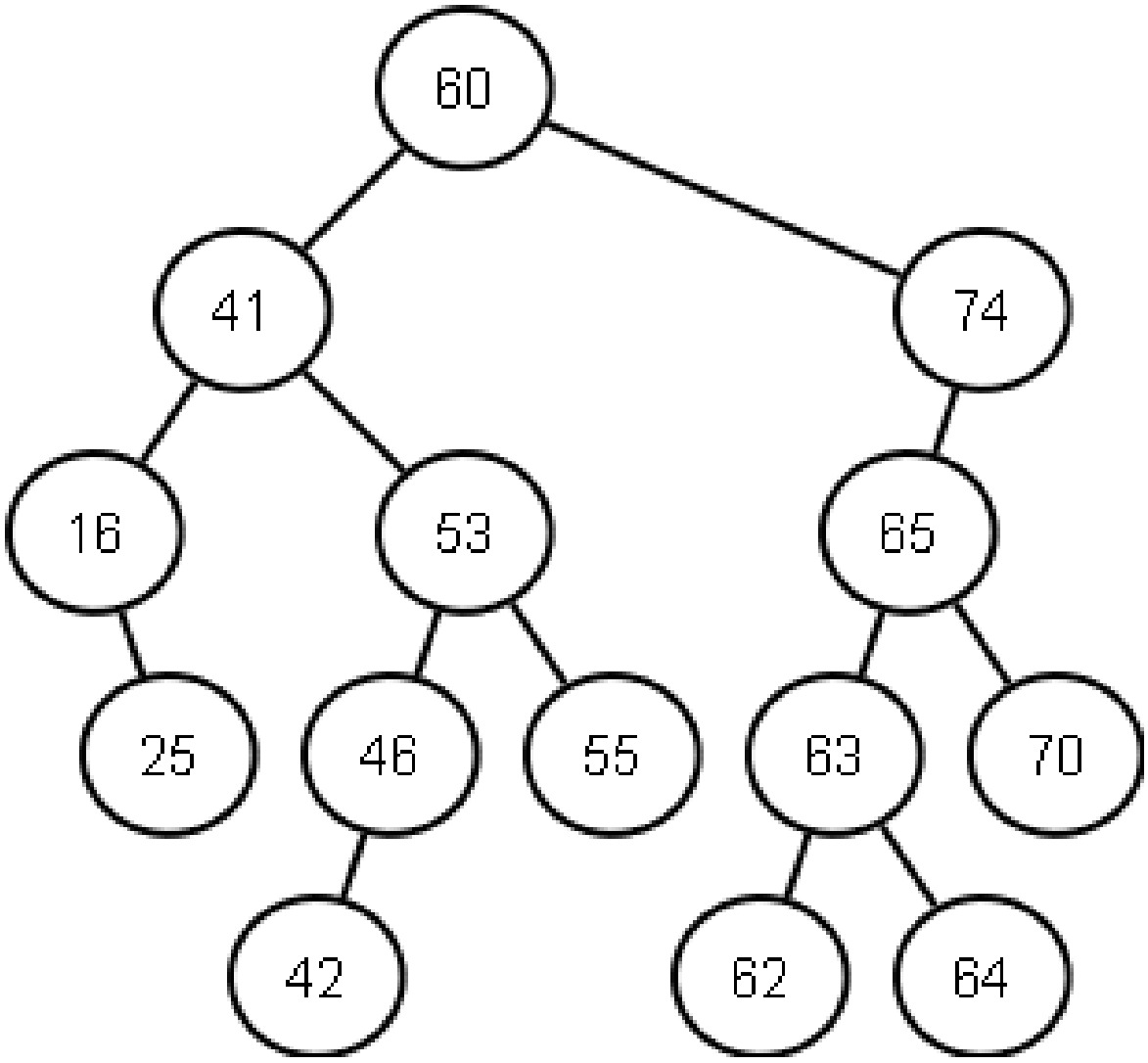
Q 04.1

How does the pivot play a role in the speed of the quick sort algorithm?

Lab 10 Computer Science 1027B

Exercise 05 (Week 05)

Given the following binary search tree:



Answer the following questions in **E05.txt (if you would like to include an image for question 2, you can submit it as E05.<png/jpg>**:

1. What are the descendants and ancestors of the node containing ‘53’?
2. Assume you wanted to remove the node containing ‘53’. What would you do? What would the resulting tree(s) look like?
3. What are the leaf nodes in this tree?
4. What are the internal nodes in this tree?
5. Perform a BFS traversal on this tree. What is the result?
6. Perform a preorder, inorder, and postorder traversal on this tree. What is the result?
7. In a file system, managing directories and files efficiently is a common problem. Each directory can contain multiple files and subdirectories, and each subdirectory can further contain more files and subdirectories. How could this hierarchical structure be represented as a tree?