# Discussion 11
# Prelim 2 review

CS 2110, FA23

## Topics

## Data structures

## Examine the following Java class for a linked node:

```
public class Node<T> {
    private Node<T> next;
    private T data;
    public Node(T init, Node<T> nextNode) {
        data = init;
        next = nextNode;
    }
    // No other methods exist.

}
```

Complete the following tasks:

1. Create (with Java code) a chain of 3 Nodes that contain the strings “Lorem”, “Ipsum”, and “Dolor” in order.
2. Create (with Java code) a chain of 2 Nodes that point to the same String array (i.e. they reference the exact same object); the array should contain {“Lorem”, “Ipsum”, “Dolor”}.

## Explain why the following real-world data / ADT pairs would be unsuitable.

1. The items in a student’s backpack | List
2. Tasks that need to be completed for a project | Bag
3. The line to order flatbreads at Mac’s | List

## Match the following real world data to the most appropriate ADT Options ( Bag, List, Stack, Queue)

1. The previous web pages visited by a user which is used by the browser when they click the back button
2. The jobs needed to be completed by a printer

## Time complexity

What is the best case and worst case time complexity for the following? Let N denote the size of the list

1. Adding an element at a specified position in a singly linked list
2. Adding an element at a specified position in a doubly linked list
3. Getting the previous node in a singly linked list (given the current node)
4. Getting the previous node in a doubly linked list (given the current node)
5. Getting an element at a specified row and column in a table implemented as a singly linked list (M rows) of singly linked lists (up to N columns)
6. Appending an element to a fixed-capacity queue implemented with a circular array
7. Appending an element to an unbounded queue implemented with a dynamic array

## Implementation of a Stack

Using a **linked structure** approach, a **Stack** can be represented by a Node<T> field called head that is the most recent item that was added to the stack. The Node<T> class has methods data() which returns the node's data and next() which returns the node containing the item that was added before it.  An empty stack has a null head.

- Implement the pop operation `pop()` which removes the node at the top of the stack and returns that node's data as a result. Throws an EmptyStackException if the stack is empty.

Imagine you are given the following iterator:

**Iterator<Integer> iterator = collection.iterator();**

Write a while-loop to print out every element of the collection on a new line.

## Efficiency

## Big Oh Notation

- Show that $5x^2+2x+1$ is in $O(n^2)$
- Show that $10+10x$ is in $O(n)$
- Show that $x+5$ is in $O(n^2)$

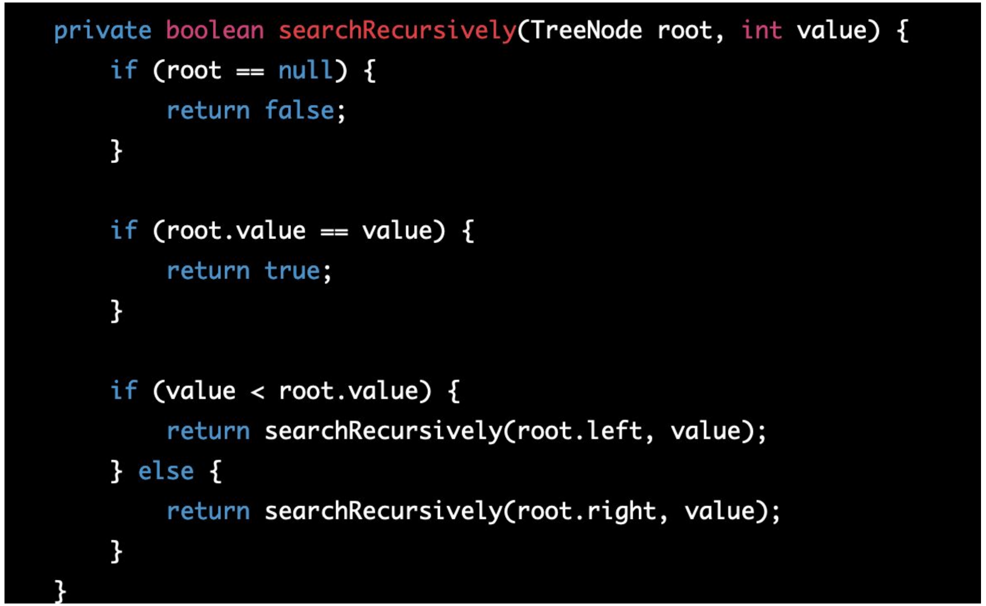
## Given the following problems state what quantity describes the problem’s size and state the algorithm’s worst case time complexity (in terms of that size) in Big Oh notation

1. Computing the mean of an array of integers
2. For some Set, enumerate every subset of size 2
3. Finding a particular value in a binary search tree
4. Finding a particular value in a sorted array using binary search

## Analyzing algorithm

Consider some BST class, and assume it has method add(), which adds nodes to the tree in BST format.

Consider the search algorithm shown. What is the best and worst time complexities of this method, in terms of the size of the tree?



```java
private boolean searchRecursively(TreeNode root, int value) {
    if (root == null) {
        return false;
    }

    if (root.value == value) {
        return true;
    }

    if (value < root.value) {
        return searchRecursively(root.left, value);
    } else {
        return searchRecursively(root.right, value);
    }
}
```

## Recursion

## Sequence of Recursive Calls

Given a recursive function and some argument values, write the sequence of calls to the function and the output it returns from each call.

```
/**returns the nth fibonacci number, where fib(0) and fib(1) are 1
 * Precondition: n must be positive
 * */
public static int fib(int n){
    if(n==0 || n == 1) {
        return 1;
    }
    else{
        return fib(n-1) + fib(n-2);
    }
}
```

What is the sequence of calls for:

fib(0)

fib(2)

fib(4)

## Solution

1) fib(0)=1
2) fib(2), fib(1)=1, fib(0)=1

   =2
3) 
```
fib(4), fib(3), fib(2), fib(1)=1, fib(0)=1, fib(1)=1, fib(2), fib(1)=1, fib(0)=1
                =2                                       =2
        =3
=5
```

## Implementing a recursive function: `tacoCat()`

Recursively implement the following method according to its specification:

```
/**
  * Returns the tacocat-version of input String [s]. A tacocat-version of
  * String [s] is [s] concatenated with the reverse of [s], but without the
  * resulting middle character duplicated.
  *
  * Ex: tacoCat(“taco”) ⇒ “tacocat”
  *     tacoCat(“race”) ⇒ “racecar”
  *     tacoCat(“kay”)  ⇒ “kayak”
  */
public static String tacoCat(String s) {

    // Your implementation here…

}
```

Author: Justin Guo (jjg283)

## Space Complexity of a Recursive Algorithm

For example consider the Fibonacci function:

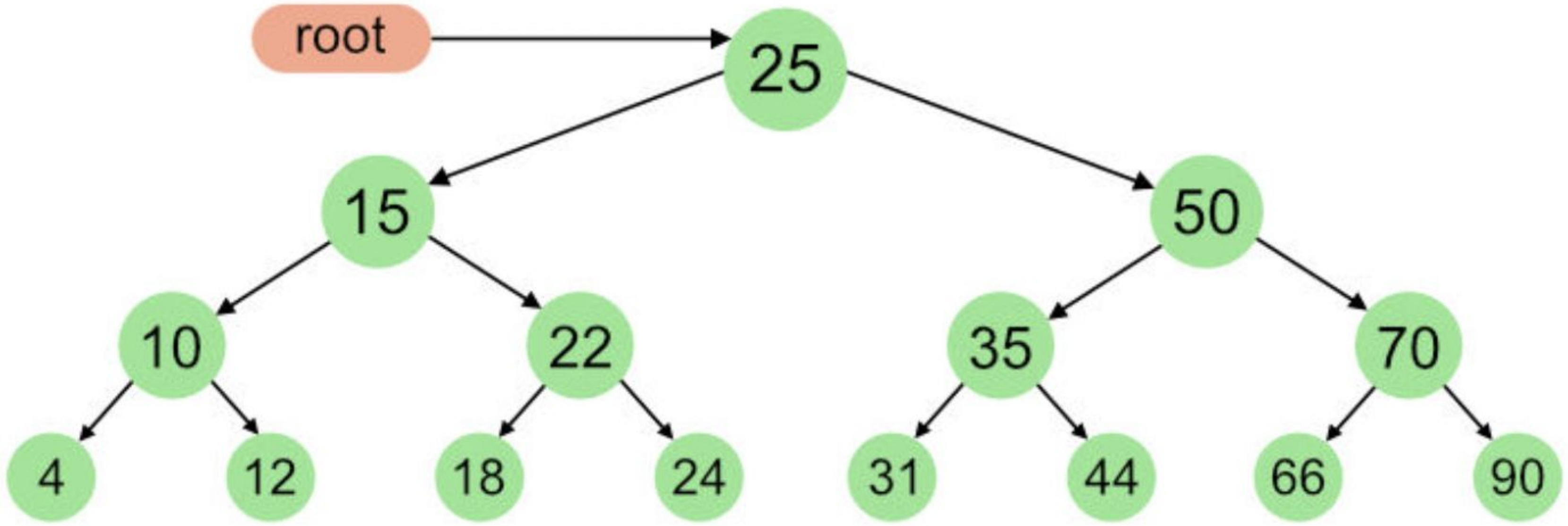
```
int fib(int n) {
      if (n <= 1) {
            // base case
            return 1;
      } else {
            // recursive case
            return fib(n - 1) + fib(n - 2);
      }
}
```

State its **space complexity** in Big O notation as a function of its parameters (n).

## Trees

# Benjamin Tang: Preorder, Inorder, Postorder

Give a preorder, inorder and postorder traversal of this graph.



Example taken from: https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/#

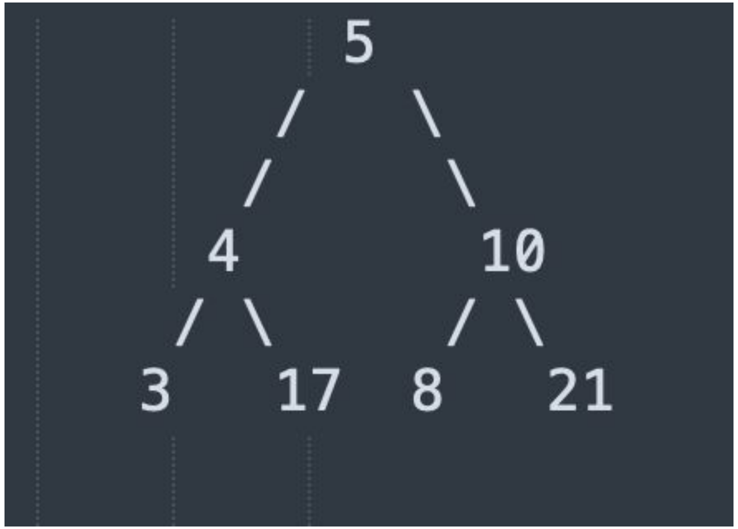
## Big Oh notation

What is the worst-case time complexity for the following, given a BST of n nodes?

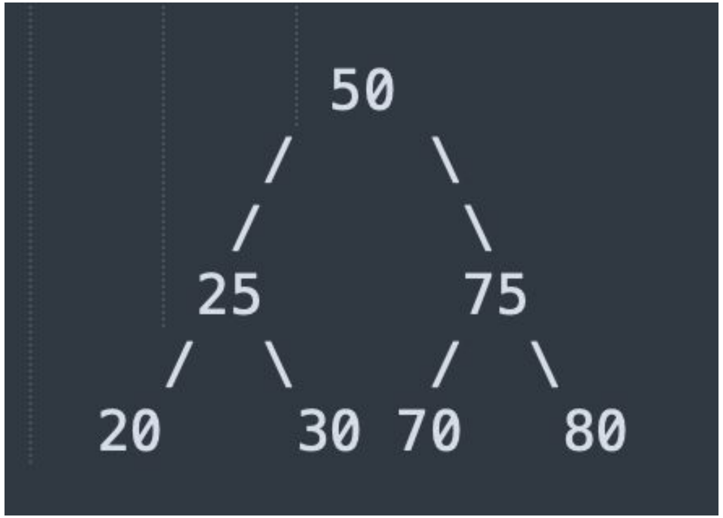
Checking if a number is in the BST

Computing the depth of the BST

## Binary Trees and Binary Search Trees



```
          5
        /   \
       /     \
      4       10
     / \     / \
    3   17  8   21
```



```
           50
         /     \
        /       \
      25         75
     /  \       /  \
   20    30   70    80
```

For both of these trees, state whether each of them are a:

- Binary Tree
- Binary Search Tree

## Loop invariants & searching

## State the numbers that are contained in the sequences below given the following range notation

a is an array in the examples below

[1,2]

[1,2)

(2,2)

(1,4]

a[1..4]

a[....4)

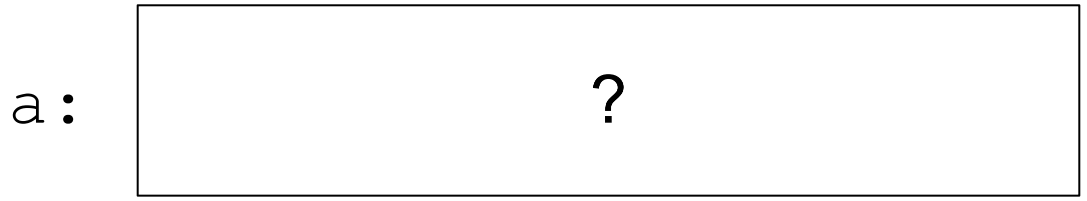
a[....4]

a(1…]

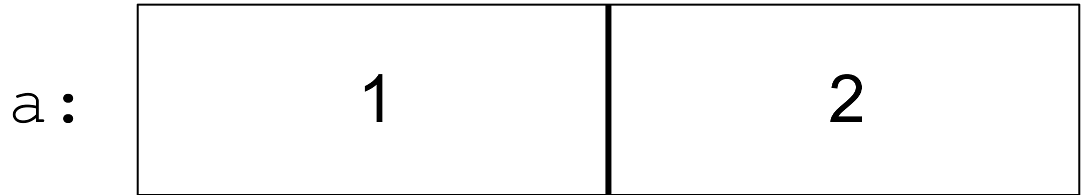
## Creating Loop Invariants from Preconditions and Postconditions

The array `a`, declared as `int[] a`, contains elements that are either `1` or `2`. Given the precondition and postcondition shown below, draw a loop invariant for this array.

Precondition

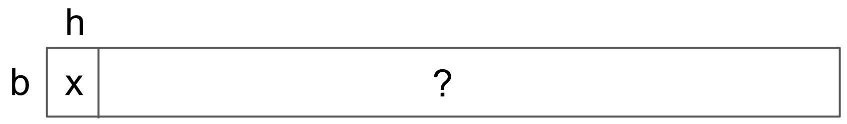


Postcondition

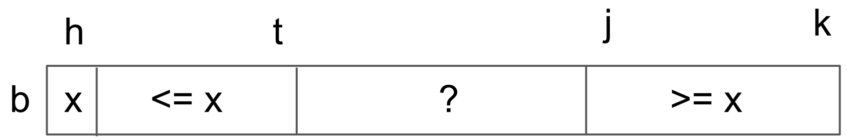


# Fill in the loop

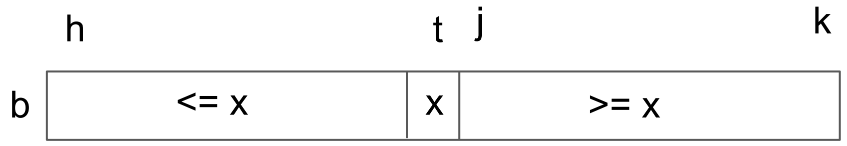
Precondition:



Loop Invariant:



Postcondition (**after** post-loop swap):



Note: this is the *method* postcondition, not the *loop* postcondition (it is not consistent with the invariant).

Fill in the missing parts (1-8) of the following code to make the loop correct :

```
public static void partition(int[] b, int h, int k){
   int t = (1)_______;
   int j = (2)_______;
   int x = b[h];

   while(t < (3)_____){
       if (b[(4)______] <= x){
           t++;
       } else {
           swap(b, (5)____, (6)____);
           j--;
       }
   }
   swap(b, (7)____, (8)____);
}
```

## Sorting & comparisons

State the worst-case *time complexities* (in terms of the number of items to be sorted, N) for:

- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort

## What is the worst case *space complexity* for the following, in terms of the number of items to be sorted, N?

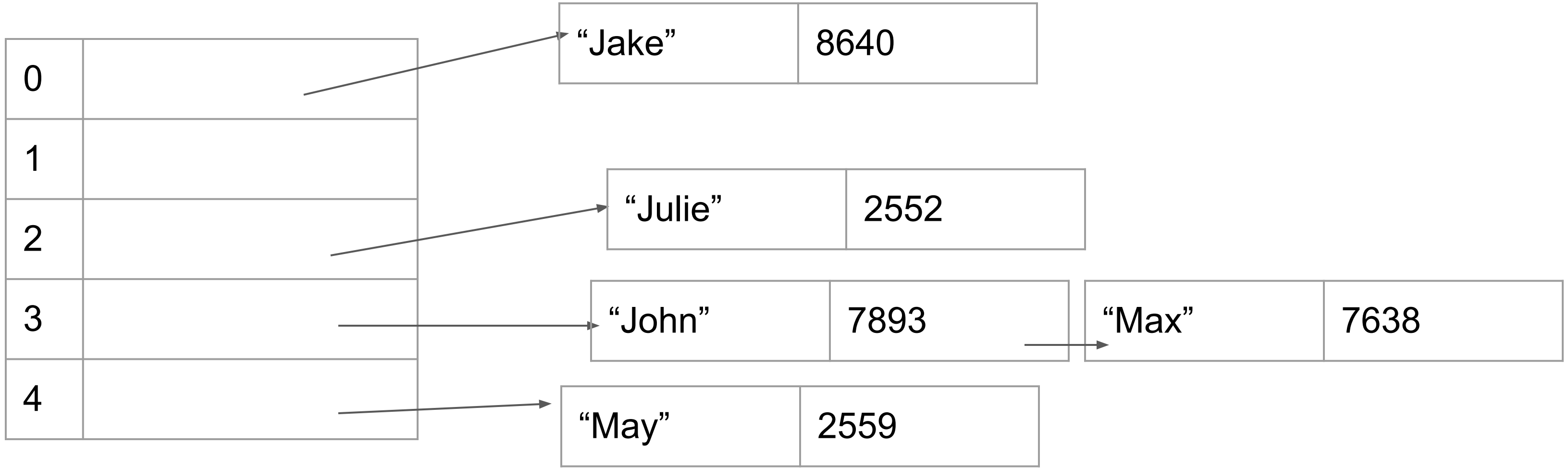
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort

## Give the tightest bound for worst-case time complexity of the following:

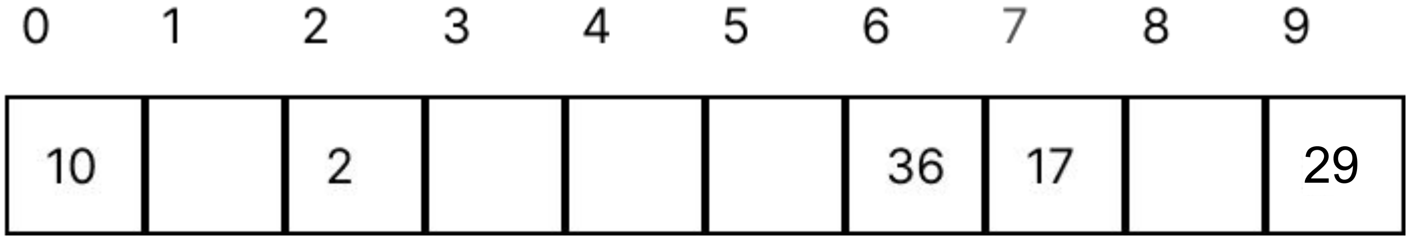
1. Merge sort on an ArrayList of size n^2
2. Quick sort on an ArrayList of size sqrt(n)
3. Insertion sort on a LinkedList of size n log n

## Dictionaries & hashing

Draw the following hashtable (representing a Set) after it is resized to have a length of 9. The names are the keys, and the numbers are the hashes associated with each key.

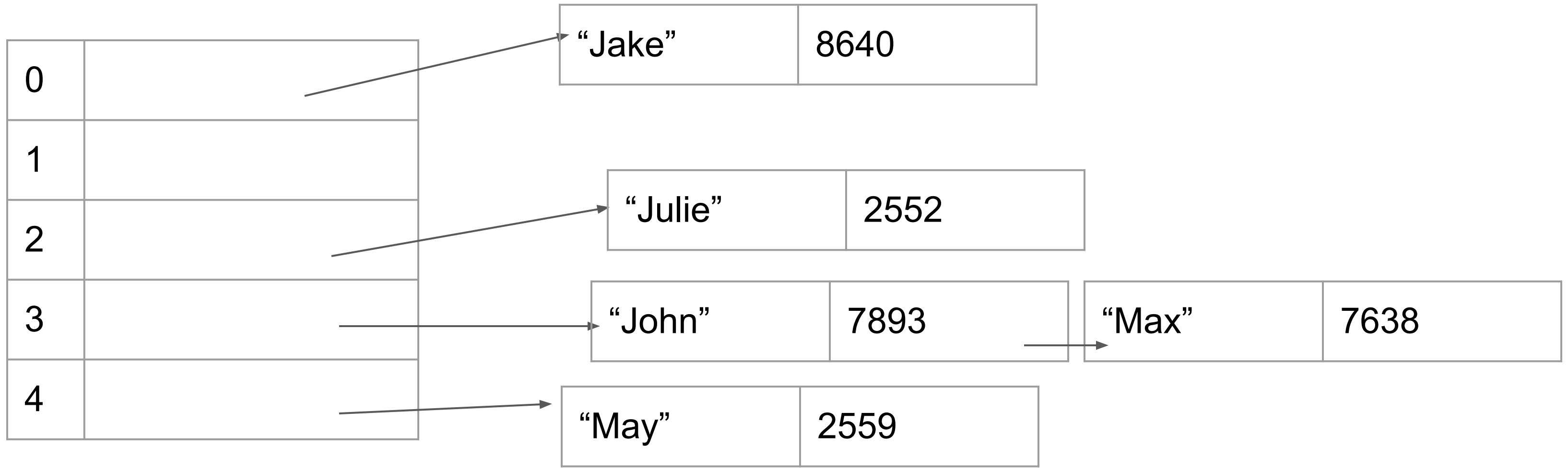


What is the index of a key with hash code 26 when inserted into this hash table using linear probing?



## Load Factors

Compute the load factor of the HT below. And what should the new bucket size be if we want the load factor to be 0.75



## GUIs & lambda expressions

Identify the event object, event source, and listener for handling button clicks.

```
class App extends JFrame
   implements ActionListener {

  App() {

    JButton b = new JButton("B");

    add(b);

    b.addActionListener(this);

  }

  @Override

  public void actionPerformed(
   ActionEvent e) {

    print("Prelim " + e.getSource());

  }

}
```

## EDT Thread

An operation needed for a program takes 10 minutes to run, would it be appropriate to execute the operation on Swing’s event dispatch thread?

## Concurrency

Three cats are at a table. Each cat cannot start eating until they have two chopsticks. They acquire one chopstick at a time and do not put them down until they have eaten. (The cats know how to use chopsticks and they refuse to eat without them.)

a) If there are 4 chopsticks, is it possible for all of the cats to starve?
b) If there are 3 chopsticks, is it possible for all of the cats to starve?

What computer science term describes the situation in which all cats starve?

## Execution of Two Threads

**Assume that the initial value of z is 3 and consider the code below in two threads. What are the possible values of z after execution of the two threads?**

**Thread 1:**

```
z =z*2
```

**Thread 2:**

```
z = z+5
```

## Deadlock

- Given two sequences of instructions (in pseudocode or Java) that are executed concurrently and require exclusive access to one or more shared resources, state whether deadlock is possible. If so, propose an alteration to one of the sequences that eliminates this possibility while preserving the desired behavior of the original instructions.

Process 1:

1) Acquire X
2) Use X
3) Release X
4) Acquire Y
5) Acquire Z
6) Use Y and Z
7) Release Y
8) Release Z

Process 2:

1) Acquire X
2) Use X
3) Release X
4) Acquire Z
5) Acquire Y
6) Use Y and Z
7) Release Y
8) Release Z

Given some variables X, Y, and Z on these two processes:

- Can deadlock occur if these processes happen simultaneously?
- If so, where, and what change can be made to avoid deadlock?