Data Structures Make-up Quizzes (20 minutes)

Name: _____ NetID: _____

By participating in this quiz, you agree to adhere to the honor code.

1. Imagine we created `IntegerDoublyLinkedList`, a class with all the same attributes and methods as `DoublyLinkedList`, but with elements that must be integers. Recall that the attributes of `DoublyLinkedList` are `Node header`, `Node trailer`, `int size`, and the methods of `Node` are `Node getNext()`, `Node getPrev()`, `void setNext(Node node)`, `void setPrev(Node node)`.

   A. Write a method for `IntegerDoublyLinkedList`, `public void switchMinMax()` that finds the nodes with the largest and smallest integer and switches their position in the list. If the list is empty, do nothing. Avoid using the methods `remove` and `addBetween` – explicitly change the `prev` and `next` attributes of nodes yourself. (8 points)

   B. If N is the size of the list, what is the worst case asymptotic complexity of this method? Briefly explain your reasoning. (2 points)

2. In a tree, multiple positions can hold the same data. Write a method for `LinkedTree<E> public int maxHeight(E element)` which returns the height of the highest node that stores element. If the element is not in the tree, return -1. Your method should run in $O(n)$ time.

   We will not take marks off for unsafe casting. Recall `Position<E>` has one method `public E getElement()` and `LinkedTree` has attributes `Node root` and `int size` and public methods `Position<E> parent(Position<E> p)`, `Goodlist< Position<E> > children(Position<E> p)`, as well as `IsInternal`, `IsExternal`, `IsRoot`, `numChildren`.

3. Ask Charlie to draw a different binary tree with nodes labelled 1-20 and answer the following questions:

   A. Write down the preorder, post order, and in order traversal of this tree.

   B. Write the nodes in the order they are visited by depth-first and breadth-first search? You can point to an answer from question 1 instead of rewriting the nodes if the order is the same.

4. A. Given a list "list" of Strings, return three lists, "uniques", "counts", "inverse" of every unique word in "list", the number of times each word appears in "list", and list of the indices of each appearance in "list".

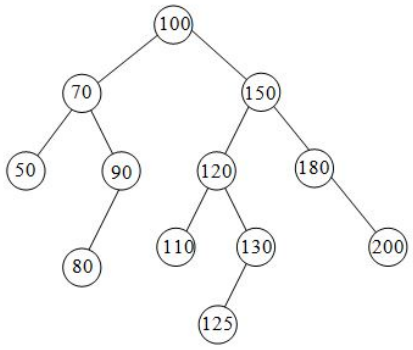
   You can assume access to an implementation of Map, "FastMap" such that `get`, `put`, and `remove` are $O(1)$.

   B. What is the complexity of your method?

5. (a) Using the simple BST rules, add 7, 3, 4, 10, 8, 5, 2, 1, 11, 12, 9 (4 points). Then remove 3 and 10 (2 points). For each addition and removal, redraw the tree. Show that the resulting tree is balanced (2 points). Add 6 using AVL rules (2 points).

6. For this week's quiz, ask Charlie to give 11 additions for a (2,4) Tree and delete 50 from the following AVL tree; show the tree before and after each restructuring.

   Consider the following AVL tree



7. Imagine we have implemented `IntAVLSearchTree` which stores entries with integer values in nodes. The nodes have attributes `Node parent`, `Node left`, `Node right`, `Entry<V> element` (the key is int!), and int height (we keep track of their height), with getters and setters for each. Imagine we just performed `put(K, V)`: 1) we add an entry, lets call it X, as in a normal binary tree, 2) we update the heights of all the nodes between X and the root, 3) we find the first unbalanced node between X and the root, let's call it A. On the path from A to X, A has descendants B then C. We now want to rearrange A, B and C to re-balance the tree using the AVL rules

   A. Write code that determines which of these three nodes should become the parent.

   B. Write code that determines which of the remaining two nodes should be the left and right children of this parent.

   C. Draw a A, B, and C in a position such that after rearrangement, B is the parent, A is the left child, and B is the right child.

8. Write an in-place algorithm implementing `int[] sortAboveThresh(int below, int above, int[] array)` that takes in a possibly unsorted integer array and returns an array in which all entries smaller than the integer above and larger than the integer below are in a sorted order (they may be adjacent to each other or not). Your implementation should be such that if the difference between above and below is smaller, the algorithm is more efficient, i.e., do not just sort array and then return everything below thresh.

9. Add 7, 3, 4, 10, 8, 5, 2, 1, 11, 12, 9 to a heap. Then remove all numbers. Draw each step.

10. (a) Say u and v are two `Vertex<V>` objects in `Graph<V,K>` graph that you know are not adjacent. Write a method `boolean commonParent(Vertex<V> u, Vertex<V> v)` that determines if there is a node that is the parent of both u to v. Your method should avoid calling `getIncoming` or `getOutgoing` more than twice.

    (b) If n is the number of vertices, m is the number of edges, and d is the maximum degree, what is the complexity of your algorithm in the case of adjacency map implementation? Bonus (1 point): what is the complexity in the two other implementations (Edge List and Adjacency Matrix)?

11. Write a method `static <V> int SearchAFollowsB(Vertex<V> start, Vertex<V> end, double below, double above, graph<V, double> myGraph)` that returns the length of the shortest directed path from start to end in graph made up only of edges with elements between `doubles above` and `below`. Return -1 if there is no such path.