CSCI 102, Spring 2025, midterm

Write your name and netID at the top of the page. Throughout you may assume you have access to `LinkedStack<E>` and `LinkedQueue<E>` which are implementations of `Stack<E>` and `Queue<E>`. This midterm is graded out of 13 points. You may use the backs of pages as scratch but please try to keep your answers on the front.

1. (1 point) Write a method for `DoublyLinkedList<E>`, `public void splice(Position<E> pos1, Position<E> pos2)` that removes all positions between `pos1` and `pos2`. Recall the method `private Node<E> validate(Position<E> pos)` and recall that the attributes of `Node<E>` are `next, prev, element`. I suggest drawing a picture. Your method doesn't have to be the most efficient, but it should not run in over $\mathcal{O}(n)$ time, where n is the number of elements in the list.

(2 points) Make sure your code works regardless of whether `pos1` comes earlier or later in the list than `pos2`; make sure to also handle the case when `pos1 == pos2`. (There are many possible solutions). You may assume `pos1` and `pos2` are part of the same (instance) list.

(2 points) Make sure you properly update `size`.

2. (2 points) Implement a method `public static <E> void transfer(Stack<E> source, Stack<E> target)` that moves all elements from `stack` to `target` so that the elements in `target` are **the same order** as they were in `source`. You may initialize other stacks and queues.

(3 points) Implement a method `public static <E> void interleave(Queue<E> queue)` that rearranges elements in a queue so that the first half is interleaved with the second half. For example, if `queue` is [1, 2, 3, 4, 5, 6], then `interleave(queue)` is [1, 4, 2, 5, 3, 6]. You may initialize **only one** other stack or queue. You may assume the queue has an even number of elements.

3. Say I give you a `BinaryTree<Integer>` called `tree`. Set $X_0 = 0$ and `pos=tree.root()`. At each step set $X_{n+1} = X_n +$ `pos.getElement()` and update `pos = tree.left(pos)` if X_n is even and `pos = tree.right(pos)` if X_n is odd. Assume that the tree is big enough so that `pos` isn't `null`.

(1 point) Draw a `BinaryTree<Integer>` of depth 4 and calculate X_0, X_1, X_2, X_3, X_4.

(4 points) Write a **recursive** method that calculates this X_n, `public static Integer x(int n, BinaryTree<Integer> tree)`.

4. (4 points) Write a method `public Position<E> deepestNode()` for `LinkedBinaryTree<E>` that returns the deepest node in the tree. **Your method must be implemented recursively!** Hint: Also return the depth of the deepest node as an auxiliary variable!

(1 point) If n is the number of elements in the tree, d(`pos`) is the depth of a position `pos`, and h(`pos`) is the height of `pos`, what is the asymptotic computational complexity of your method in big O notation?