CSCI 1933 Lab 14

Final Review

Rules

You may work individually or with *one* other partner on this lab. All steps should be completed primarily in the lab time; however, we require that at least the starred steps (★) to be completed and checked off by a TA during lab time. Starred steps cannot be checked off during office hours. You have **until the last office hours on Friday, May 5th** to have non-starred steps checked by ANY TA, but we suggest you have them checked before then to avoid crowded office hours on Monday. There is nothing to submit via Moodle for this lab. This policy applies to all labs.

NOTE: You cannot use computers to write code for this lab.

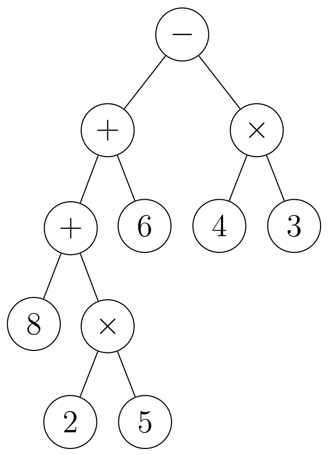
1 Review Questions ★

(a) What are the Big O runtimes for enqueing and dequeing in a Priority Queue for both node and array representations? Explain.

(b) If a binary search tree is full, which implementation would be more efficient: an array or linked node?

(c) What is the difference between `throw` and `throws` in Java?

(d) Write a pre-order, in-order, and post-order node traversal of the following binary tree. Explain.



2 Find Range ★

Given a sorted array of integers, write a function that returns all the values in the array that are greater than equal to x and less than equal to y. Explain the runtime of your code.

Ex: arr = [2, 5, 6, 8, 10]; x = 3, y = 8 Sol: [5, 6, 8]

```
public int[] numbersInRange(int[] arr, int x, int y) {
        // TODO: Complete this code.
}
```

3 LinkedList ★

Given a linked-list of double, write a function `public double findMiddle(Node head)` that find and returns the middle element in the list.

```
  public class Node {

    private Node next;
    private double data;

    public Node(){} //default constructor

    public Node(double n, Node link){ //constructor
      data = n;
      next = link;
    }

    public Node getNext(){ //selector
      return next;
    }

    public double getData(){ //selector
      return data;
    }

    public void setNext(Node link){ //mutator
      next = link;
    }
    public void setData(double n){ //mutator
      data = n;
    }

  }
  public static double findMiddle(Node head) {
          // TODO: Complete this code.
  }
```

4 Tree Traversal

Complete the method `isSubtree()` as part of the BTNode class below. The method should return true if a similar substree (T) is found to be identical in structure and value within the main tree.

```
public class BTNode<T extends Comparable<T>>{
        private T data;
        private BTNode<T> left;
        private BTNode<T> right;

        public BTNode<T> getLeft() { return left; }
        public BTNode<T> getRight() { return right; }
        public T getData() { return data; }

        public void setLeft(BTNode<T> l) { left = l; }
        public void setRight(BTNode<T> r) { right = r; }
        public void setData(T d) { data = d; }

        public boolean isSubTree(BTNode<T> o){
                // TODO: Complete this code.
        }
```