Q0. You are to find top K largest items from an unsorted array of size N. Your algorithm must be in-place. How would you achieve this using a heap? Explain your algorithm. (4 pts) What is the time complexity? (4 pts)

Q1. True/False. Any AVL can be converted into a red-black tree without a structural change. Justify your answer. If true, provide an algorithm to do so. If false, provide a counter example. (4 pts)

Q2. True/False. If a binary tree is colored such that the shortest root-to-leaf path consists entirely of black nodes, and the longest path alternates between black and red nodes (with the root being black), and the longest path is exactly twice as long as the shortest path, then the tree is guaranteed to satisfy all red-black tree properties. Justify your answer. If true, explain why. If false, provide a counterexample.

Q3. You are repeatedly inserting new **red nodes** into a red-black tree that currently has **n nodes** and **black height = b**. We want to know how many insertions are required before the tree's black height increases to **b + 1**.

1. What is the **best case** (fewest insertions needed)?
2. What is the **worst case** (most insertions needed)?

Justify your answers clearly with an example for both cases.

Q4. Insert the integers **1 through 11** into an initially empty **red-black tree**, in **ascending order**.

1. Draw the **intermediate trees as well as the final tree** showing all node colors.
2. Indicate the **rotations and recolorings** that occur during the insertion process.

Q5. (True/False) Inserting the same set of keys into a red-black tree in different orders can result in different valid red-black trees. If true, provide an example. If false, explain why the resulting tree must be unique.

Q6. (True/False) Given a red-black tree, we can uniquely determine the insertion order of its keys. If true, devise an algorithm. If false, provide a counter example.

Q7. Bonus Question. Binary tree and doubly linked list.

Both binary trees and doubly linked lists contain nodes that have references to two other nodes. Write code that transforms a binary tree into a doubly linked list. The order of nodes in the doubly linked list must be same as <u>In-order traversal</u> of the given Binary Tree.

```
//class Node is given here (MUST NOT change this class)
public class Node<E> {
    public E data;     // data stored at this Node<E>
    public Node left;  //reference to left subtree (or prev in doubly linked list)
    public Node right; // reference to right subtree (or next in doubly linked list)

    public Node(E data, Node left, Node right) {
        this.data = data;
        this.left = left;
        this.right = right;
    }
}
```

(1) Implement `transform` method.

```
//overallRoot is the root of the binary tree to convert
public static Node transform(Node overallRoot);
```

(2) What is the time complexity of your implementation? Explain your answer

(3) What is the space complexity of your implementation? Explain your answer

Scratch Paper