1. Consider an $n$-node treap $T$. As in the lecture notes, we identify nodes in $T$ by the ranks of their search keys; for example, ‘node 5’ means the node with the 5th smallest search key. Let $i, j, k$ be integers such that $1 \leq i \leq j \leq k \leq n$.

(a) What is the exact probability that node $j$ is a common ancestor of node $i$ and node $k$?
(b) What is the exact expected length of the unique path in $T$ from node $i$ to node $k$?

Don’t forget to prove that your answers are correct!

2. A meldable priority queue stores a set of keys from some totally-ordered universe (such as the integers) and supports the following operations:

- **MAKEQUE**: Return a new priority queue containing the empty set.
- **FINDMIN(Q)**: Return the smallest element of $Q$ (if any).
- **DELETEMIN(Q)**: Remove the smallest element in $Q$ (if any).
- **INSERT(Q, x)**: Insert element $x$ into $Q$, if it is not already there.
- **DECREASEKEY(Q, x, y)**: Replace an element $x \in Q$ with a smaller key $y$. (If $y > x$, the operation fails.) The input is a pointer directly to the node in $Q$ containing $x$.
- **DELETE(Q, x)**: Delete the element $x \in Q$. The input is a pointer directly to the node in $Q$ containing $x$.
- **MELD(Q_1, Q_2)**: Return a new priority queue containing all the elements of $Q_1$ and $Q_2$; this operation destroys $Q_1$ and $Q_2$.

A simple way to implement such a data structure is to use a heap-ordered binary tree, where each node stores a key, along with pointers to its parent and two children. **MELD** can be implemented using the following randomized algorithm:

```
MELD(Q_1, Q_2):
    if Q_1 is empty return Q_2
    if Q_2 is empty return Q_1
    if key(Q_1) > key(Q_2)
        swap Q_1 \leftrightarrow Q_2
    with probability 1/2
        left(Q_1) \leftarrow MELD(left(Q_1), Q_2)
    else
        right(Q_1) \leftarrow MELD(right(Q_1), Q_2)
    return Q_1
```

(a) Prove that for any heap-ordered binary trees $Q_1$ and $Q_2$ (not just those constructed by the operations listed above), the expected running time of **MELD(Q_1, Q_2)** is $O(\log n)$, where $n = |Q_1| + |Q_2|$. [Hint: How long is a random root-to-leaf path in an $n$-node binary tree if each left/right choice is made with equal probability?]

(b) Show that each of the other meldable priority queue operations can be implemented with at most one call to **MELD** and $O(1)$ additional time. (This implies that every operation takes $O(\log n)$ expected time.)