1. Prove the following useful property of a binary search tree (with distinct keys):

**Property 1.** Let \( x \) be a node in a BST \( T \). Let \( \text{max} \) and \( \text{min} \) denote the largest and smallest keys in the subtree rooted at \( x \), respectively. For any node \( y \) outside the subtree rooted at \( x \), show that

\[
\text{either } y.\text{key} > \text{max} \quad \text{or} \quad y.\text{key} < \text{min}
\]

This implies that if there is a key \( k \) in the tree that satisfies \( \text{min} \leq k \leq \text{max} \) then it must lie inside the subtree rooted at \( x \). (Here the subtree rooted at \( x \) includes \( x \) itself.)

Use it to solve Exercise 12.2-5, 12.2-6 and 12.2-9 on page 293. In all three exercises, we assume the BST has distinct keys.

2. Problem 13-2: Join operation. For a): you only need to answer the following two questions: 1) Let \( T \) be a red-black tree in which the root has black height \( T.\text{bh} \). Then after an insertion, \( T.\text{bh} \) either stays the same or increases by 1. Describe the scenario when it increases by 1. 2) If a node \( z \) has black height \( h \), use \( O(1) \) time to compute the black height of \( z \)'s children. Skip e) and f). Replace d) by the following: If \( T_1.\text{bh} = T_2.\text{bh} \), what color should we make \( x \) to get a red-black tree? If \( T_1.\text{bh} > T_2.\text{bh} \), what color should we make \( x \) so that properties 1, 2, 3 and 5 are maintained? “Briefly” describe how to enforce property 4 in \( O(\lg n) \) time.

3. Exercise 14.1-8 on page 345. (Hint: Each chord \( c = (c_1, c_2) \) has two endpoints \( c_1, c_2 \in [0, 2\pi) \), where \( c_1 < c_2 \). Given two chords \( c = (c_1, c_2) \) and \( c' = (c'_1, c'_2) \), how do we determine they intersect inside the circle or not? Also if \( c \) is one of the shortest chords, how many chords does \( c \) intersect?)

4. Exercise 14.3-6 on page 354. You only need to describe the following key points: 1). What extra information to store in each node? 2). With this additional information in each node, how to answer Min-GAP efficiently? 3). Use Theorem 14.1 to prove that insertion and deletion of a node can still be done in \( O(\lg n) \) time: Show that all the extra information for a node \( x \) can be derived from the information stored in its two children in \( O(1) \) time.

5. Exercise 16.1-5 on page 422 and Exercise 16.3-5 on page 436. (Hint for 16.1-5: You may not want to be greedy here.)