A. RECAP: Binary Search Trees and Balanced Search Trees

Your preceptor will briefly review key points of this week's lectures.

B. EXERCISE (Midterm Review): Subtree Counts (Spring '21 Midterm)

Design a *multiset* data type that supports adding integer keys and performing the following two types of queries:

- **Count**: Given an integer \( k \), determine the number of integers in the multiset equal to \( k \).
- **Rank**: Given an integer \( k \), determine the number of integers in the multiset strictly less than \( k \).

To do so, implement this API:

```java
public class Multiset
{
    Multiset() // create an empty multiset
    void add(long k) // add the integer k to the multiset
    int count(long k) // number of integers in the multiset equal to k
    int rank(long k) // number of integers in the multiset strictly less than k
}
```

Note that, unlike in a symbol table with integer keys, an integer can appear in a multiset more than once. Here is an example:

```java
Multiset multiset = new Multiset(); // [ ]
multiset.add(20); // [ 20 ]
multiset.add(30); // [ 20 30 ]
multiset.add(40); // [ 20 30 40 ]
multiset.add(30); // [ 20 30 30 40 ]
multiset.add(30); // [ 20 30 30 30 40 ]
multiset.add(20); // [ 20 20 30 30 30 40 ]
multiset.count(30); // 3
multiset.count(35); // 0
multiset.rank(30); // 2
multiset.rank(35); // 5
```

For full credit, each operation must take \( O(\log n) \) time and use \( O(n) \) extra space, where \( n \) is the number of integers added to the multiset. Half credit if \( \text{rank()} \) takes \( \Theta(n) \) time.
C. EXERCISES (Midterm Review): Red-Black Trees

1) (Spring ’23 Midterm) The following BST satisfies perfect black balance, but violates color invariants:

![Red-Black Tree Diagram]

Give a sequence of 4 elementary operations (color flip, rotate left or rotate right) that restore the color invariants.
2) (Fall ’19 Midterm) Suppose that you insert the key 21 into the following left-leaning red–black BST:

```
2  6
 4  8 10
  12
```

Give the sequence of 4 elementary operations (color flips and rotations) that result.

D. EXERCISE (Midterm Review): Runtime Analysis (Fall ’17 Midterm)

Consider an array that contains two successive copies of the integers 1 through \( n \), in ascending order. For example, here is the array when \( n = 8 \):

```
1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8
```

Note that the length of the array is \( 2n \), not \( n \).

(a) How many compares does selection sort make to sort the array as a function of \( n \)? Use tilde notation to simplify your answer.

\[
\sim \quad \text{compares}
\]

(b) How many compares does insertion sort make to sort the array as a function of \( n \)? Use tilde notation to simplify your answer.
(c) How many compares does mergesort make to sort the array as a function of $n$? Assume $n$ is a power of 2. Use tilde notation to simplify your answer.

E. EXERCISE (Midterm Review): Memory Analysis (Fall ‘14 Midterm)

Suppose that you implement a left-leaning red-black BST using the following representation:

```java
public class RedBlackBST<Key extends Comparable<Key>, Value> {
    private Node root; // root of BST
    private int N; // number of key-value pairs
    private class Node {
        private Key key; // symbol table key
        private Value value; // symbol table value
        private Node left; // left child
        private Node right; // right child
        private boolean color; // color of link from parent
        private int count; // number of nodes in subtree rooted at this node
    }
    ...
}
```

Using the 64-bit memory cost model from lecture and the textbook, how much memory (in bytes) does a RedBlackBST object use as a function of the number of key-value pairs $N$? Use tilde notation to simplify your answer.

*Include all memory except for the Key and Value objects themselves (because you do not know their types).*