Problem 1-1. (Red-Black Trees)
Red-black trees are approximately balanced. A tree with n nodes is "balanced" if its height is $O(\log n)$. This guarantees that dynamic-set operations such as SEARCH, PREDECESSOR, SUCCESSOR, MINIMUM, MAXIMUM, INSERT,and DELETE run in $O(\log n)$ time. A red-black tree is a binary tree that satisfies the following red-black properties:

1. Every node is either red or black.
2. The root is black.
3. Every leaf (NIL) is black.
4. If a node is red, then both its children are black.
5. For each node, all simple paths from the node to descendant leaves contain the same number of black nodes.

(a) Suppose that a node x is inserted into a red-black tree with RB-INSERT and then is immediately removed with RB-DELETE. Is the resulting red-black tree the same as the initial red-black tree? Justify your answer.

Problem 1-2. (Compute the Levenshtein Distance)
In 1965, Vladimir Levenshtein defined the distance between two words as the minimum number of “edits” it would take to transform the misspelled word into a correct word, where a single edit is the insertion, deletion, or substitution of a single character. Given two strings, represented as arrays of characters A and B, compute the minimum number of edits needed to transform the first string into the second string.

(a) Let the Levenshtein distance between the two strings A and B be represented by $E(A,B)$. Lets say that a and b are, respectively, the length of strings A and B. Recursively define the value of the optimal solution.

Hint: Consider the same problem for $A[0:i-1]$ *and* $B[0:j-1]$.

(b) Compute the minimum number of edits to transform A into B provided the recursive definition of $E(A,B)$ found above.

Hint: Tabulate the values of $E(A[0:k],B[0:l])$. *An example E table for "Carthorse" and "Orchestra" is provided in the following Figure.*

		C	a	r	t	h	o	r	s	e
	0	1	2	3	4	5	6	7	8	9
O	1	1	2	3	4	5	6	7	8	9
r	2	2	2	2	3	4	5	6	7	8
c	3	3	3	3	3	4	5	6	7	8
h	4	4	4	4	4	3	4	5	6	7
e	5	5	5	5	5	4	4	5	6	6
s	6	6	6	6	6	5	5	5	5	6
t	7	7	7	7	6	6	6	6	6	6
r	8	8	8	7	7	7	7	6	7	7
a	9	9	8	8	8	8	8	7	7	8

(c) What is the time complexity of this algorithm?

(d) What is the the memory requirement?

Problem 1-3. (Find the Longest Nondecreasing Subsequence)
The problem of finding the longest nondecreasing subsequence in a sequence of integers has implications to many disciplines, including string matching and analyzing card games. The length of the longest nondecreasing subsequence for array A in the following Figure is 4. There are multiple longest nondecreasing subsequences, e.g. $\langle 0, 4, 10, 14 \rangle$ and $\langle 0, 2, 6, 9 \rangle$.

0	8	4	12	2	10	6	14	1	9
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Given an array A of n numbers, find a longest subsequence $\langle i_0, \ldots, i_{k-1} \rangle$ such that $i_j < i_{j+1}$ and $A[i_j] \leq A[i_{j+1}]$ for any $j \in [0, k-2]$.

Hint: Express the longest nondecreasing subsequence ending at $A[i]$ in terms of the longest nondecreasing subsequence in $A[0 : i-1]$.

(a) Write a recurrence for s_i, the length of the longest nondecreasing subsequence of A that ends at $A[i]$.

(b) We want the longest subsequence of A, not just the length of the longest subsequence. Implement a dynamic solution that relies on the recursively defined s_i. *Hint: In addition to storing a table for s_i, the length of the longest subsequence ending at $A[i]$, consider storing a table for the index of the last element of the sequence that we extended to get the current sequence.*

(c) What is the time complexity of this solution?

(d) What is the memory requirement?