Assignment 3

CS-GY 6033 INET Spring 2025

Due date:
March 23rd 2025, at 11:55pm on Gradescope.

Instructions:
Justify all your answers using techniques from class. Your solution is to be written out (or typed) and submitted online through Gradescope before the deadline.

Question 1: Binary Search trees

.

(a) 4 points. Draw the BST that results from inserting the keys in the following order

$$3, 1, 5, 4, 7, 9, 11, 15, 2, 6, 8, 10$$

Next, show the tree after each of the following deletions: $5, 11, 15, 7, 9$.

(b) 4 points. Count the number of comparisons carried out when a BST is built by inserting the keys in their given order: $3, 5, 2, 7, 1, 9, 10$. Justify how you get the total. Recall that in class, we showed that the number of comparisons carried out by the BST building process is the *same* as the number of comparisons carried out by Quicksort. Your job is to show the execution of Quicksort (and which pivots) corresponds to the BST that you built. Be sure to illustrate that both procedures use the exact same number of comparisons.

(c) 4 points Write a procedure called Delete-Min(T) that takes as input a reference to the root node of a BST, and deletes the node containing the minimum key. The procedure must return a reference to the tree after the deletion. You may *not* simply call the BST-delete procedure from class.

(d) 6 points Let T be the root node of a non-empty Binary Search tree. Consider the following algorithms:

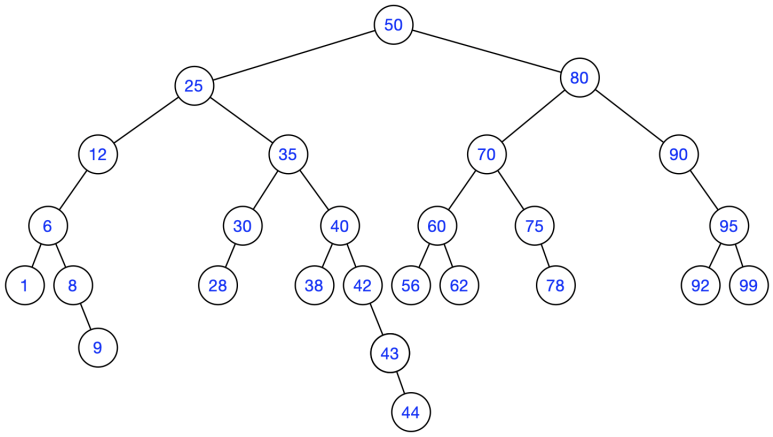
```
PrintTree1(T)
    if T.left ≠ NIL
        print T.key
    if T.left ≠ NIL
        PrintTree1(T.left)
    if T.right ≠ NIL
        PrintTree1(T.right)
```

```
PrintTree2(T)
    if T ≠ NIL
        if T.left ≠ NIL
            print T.key
            PrintTree2(T.left)
            PrintTree2(T.right)
```

```
PrintTree3(T)
    if T ≠ NIL
        if T.left ≠ NIL
            print T.key
        PrintTree2(T.left)
        PrintTree2(T.right)
```

Execute each of these algorithms on the following tree (you need only show the final output).

Next, you must determine which of these procedures are *guaranteed* to produce the equivalent output on any tree. Recall that the initial input T is a non-empty BST. Justify which procedures are equivalent and why. Justify which procedures are different and why.



Question 2: Tree traversals

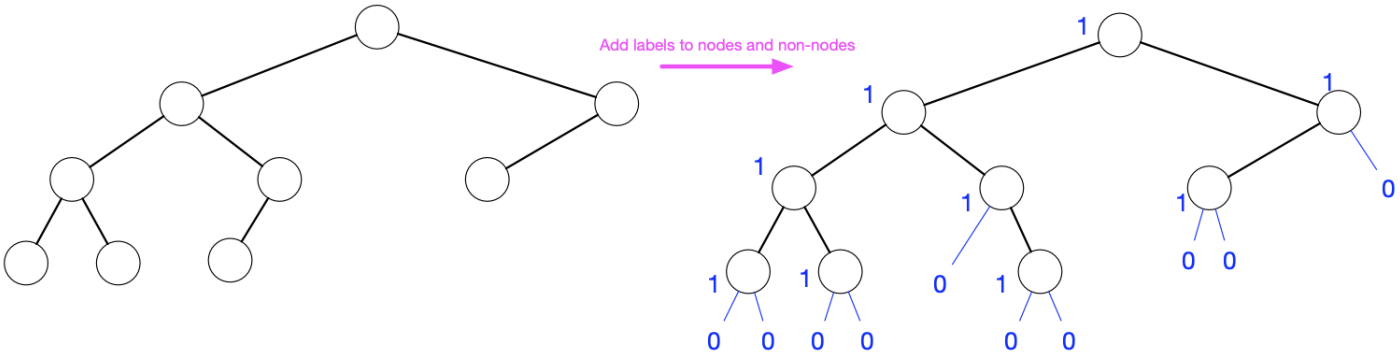
.

(a) 2 points Given an example of a BST such that the Inorder traversal and the Postorder traversal produce the same output. Your tree must consist of the nodes $3, 2, 5, 1, 6, 8$.

(b) 4 points Suppose x is a reference to a node in a BST. Write a **recursive** procedure FindDepth(x) that takes as input only the reference x (and not the root of the tree), and returns the *depth* of the node x in the tree. Recall that the root node has depth 0, and the children of the root have depth 1.

(c) 6 points Let T be a reference to the root node of a binary search tree and let x and y be references to nodes in the tree. Your job is to write a procedure that finds the lowest **ancestor** of both nodes x and y. For example, the nodes 44 and 28 in the BST from 1(d), have common ancestor 35. Call your procedure FindAncestor(T, x, y). You may assume the initial input T is non-empty.

(d) 8 points One way to encode the shape of a binary tree is using a string of 0's and 1's. The idea is to take a binary tree, and imagine that each node represents a 1, and every reference to NIL (the lack of a node) represents a 0. We then take the pre-order traversal of the tree, and the resulting string is a way to encode the shape of the binary tree. An example is shown below, where the original shape of the binary tree is shown on the left, and we add 1's and 0's as described above. The resulting pre-order traversal is: 11110010010100110000.

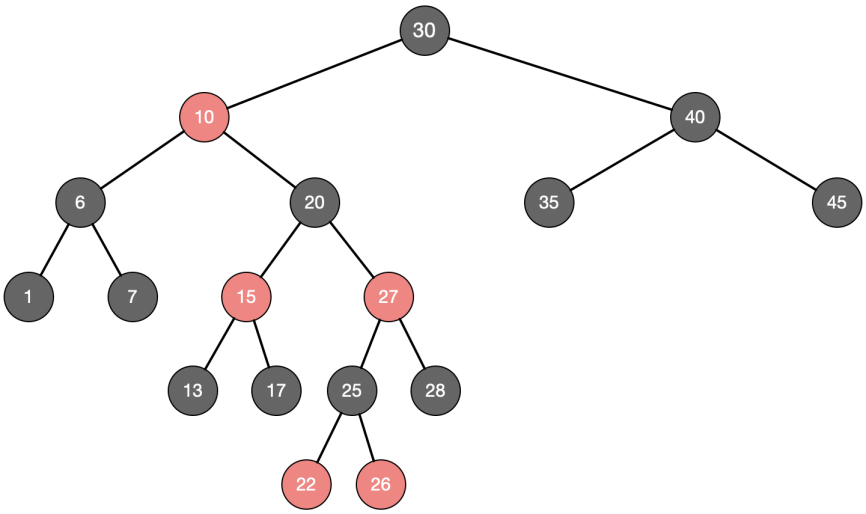


Your job is to write a recursive procedure called RecreateTree(S) where S is a *Stack* object, containing the sequence of 1's and 0's. The procedure must return a reference to the root not of the binary tree that corresponds to the string in S.

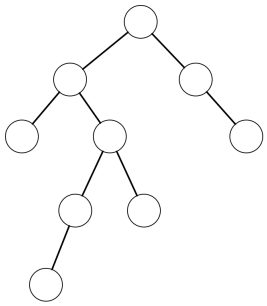
Notes: You may use newNode() for creating a new node object. You may use S.pop() to remove the top digit from the stack S. Finally, you may assume that the stack S is passed by reference to the recursive calls.

Question 3: Red-black trees

(a) 6 points Consider the red-black tree shown below. Your job is to first add all NIL nodes to the tree. Next, show how to insert the new node 23. You must show both the initial insertion, and any changes made by RB-repair. You must also show the updates to any attributes of the tree nodes. Explain briefly why the black height of the tree does not change after this insert. After 23 is inserted, is there another node that could be inserted which would cause the black-heigh to increase?



(b) 3 points Consider the BST shape below. Your job is to show how to color the nodes and add NIL leaves so that the coloring represents a valid Red-Black tree, OR, you must justify why this try cannot be correctly colored.



(c) 8 points Answer each of the following with a brief justification:

1. What is the maximum number of nodes in a red-black tree of black height 4?
2. A red-black tree has 60 nodes. Is it possible that all nodes in the tree are colored black?
3. A red-black tree has 73 nodes. Is it possible that such a tree has a black height of 3? If so, sketch such a tree.
4. A red-black tree has 15 nodes. What are the different possible black heights for such a tree? For each possibility, sketch a coloring of a tree with the specified black height.

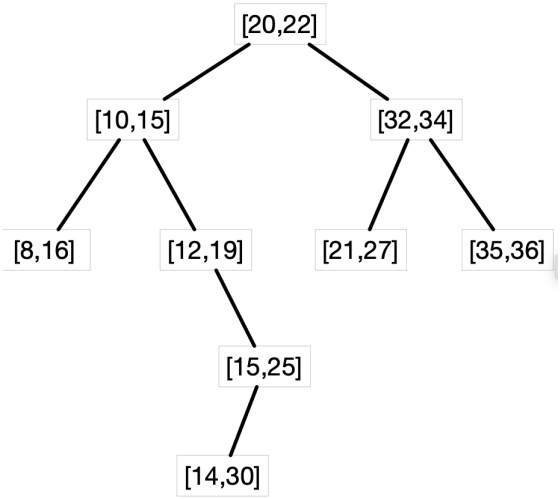
Question 4: Interval Trees

.

(a) 3 points Write a procedure to return a reference to the node in an interval which is the *last to finish*. Call your procedure FindLast(T), where T is a reference to the root of the tree.

(b) 3 points

An interval tree is drawn below. Note that the max attributes are not shown. Given an example of an interval i for which INTERVAl-SEARCH(i) returns node $[21, 27]$, or explain why it is impossible.



(c) 6 points Let T be a reference to the root node of an interval tree, where each node is augmented with $x.max$ as defined in class. Write a recursive procedure called MaxRight(T, k) which returns the maximum right endpoint out of all intervals in T which start *on or after* time k. Justify the runtime of $O(n)$

Question 5: Augmented BSTs

Let T be an interval tree that stores information on projects running in a company . Each tree node x contains the following attributes: **x.name** (the name of the project), **x.start** (the start time of the project), **x.end** (the end time), **x.budget** (the project budget), **x.max** (the usual max attribute used in interval trees), **x.size** (the size of the subtree rooted at x) and **x.btotal**, (the total of all budgets for projects in the subtree rooted at x.)

For the following questions, you must write pseudo-code. You may include in your pseudo-code any calls to procedures from class, such as BST-rank and BST-select.

(a) 6 points Write the pseudo-code for a recursive algorithm called CostAfter(T, k) which returns the total budget for *all* projects in the tree which start after time k. You must justify the runtime of $O(h)$ of your algorithm, where h is the height of your tree.

(b) 6 points Write the pseudo-code for a procedure called TotalBudget(T, k) which returns the total budget from all projects that start *before the kth project* in the set. For example, if $k = 5$, you want to find the total budget of the first, second, third and fourth projects to start. You must justify the runtime of $O(h)$ of your algorithm, where h is the height of your tree.