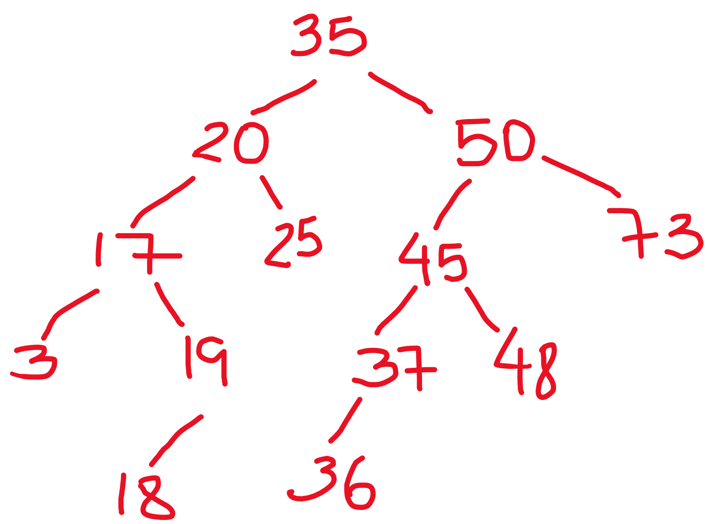
CS112 Data Structures - Midterm 2 - Fall 2022

Name: ______________________________________ NetID: ___________________________

- WRITE your **name** and **NetID** on EVERY page.
- DO NOT REMOVE the staple on your exam.
- WRITE NEATLY AND CLEARLY. If we cannot read your handwriting, you will not receive credit. Please plan your space usage. No additional paper will be given.
- This exam is worth 150 points.

Problem 1 – Binary Search Tree (BST) comprehension (28 points)

(a) **(5 points)** Build the BST whose keys are inserted in the following sequence.
35 20 25 50 17 73 45 19 37 3 18 48 36



The Following questions refer to the BST on part (a).

(b) **(3 points)** What is the tree height of the BST (the root has height 0)? 4

(c) **(3 points)** List the keys that requires 4 compares (count compareTo) for a search hit. 3, 19, 37, 48

(d) **(5 points)** What could be the maximum height of a BST built with the same keys, but with different insertion sequences, and why?
[2 points] maximum height is 13
[3 points] if the keys are inserted in a descending order, you get a BST skewed to the left, if the keys are inserted in an ascending order, you get a BST skewed to the right.

(e) **(4 points)** For the cases that the BST with the maximum height, what could be the key in the root node? 73 if the BST skewed to the left, 3 if the BST skewed to the right.
// Full credit if student responds with just one of the keys.

(f) **(5 points)** What could be the minimum height of a BST built with the same keys, but with different insertion sequences, and why?
[2 points] minimum height is 3

Name: ______________________________________ NetID: ___________________________

[3 points] There are 13 keys for the BST, each level of the tree should be full except the bottom level to have the minimum height. Maximum number of nodes for a BST with a tree height 2 is $2^{2+1} - 1 = 7$ nodes, thus, need an additional level to build a BST with 13 nodes.

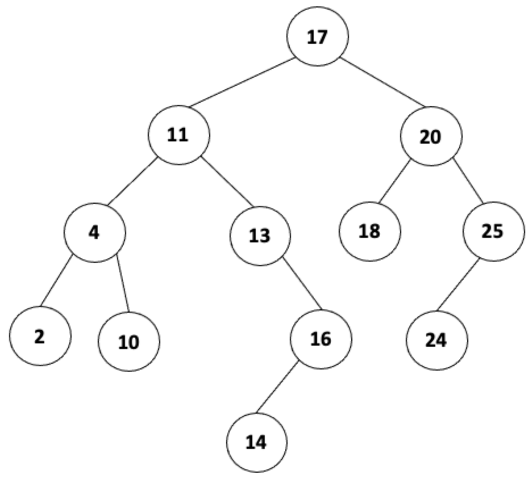
(g) **(3 points)** What is the average number of compares (count compareTo) for a search hit? Simply write the expression. (1 + 2 + 2 + 3 + 3 + 3 + 3 + 4 + 4 + 4 + 4 + 5 + 5)/13 = 43/13

Name: ______________________________________ NetID: ___________________________

Problem 2 – Binary Search Tree (BST) implementation (20 points)

(a) **(5 points)** Assume the print () operation below is provided in the BST implementation. Given the BST below, write the output when a client program calls the print() method. The root node is the node containing the key 17.

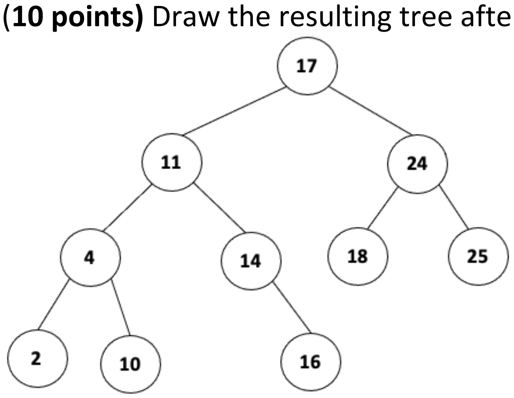
```
public void print() {
    foo(root);
}

private void foo(Node x)
{
    if (x == null)
        return;
    StdOut.print(x.key);
    foo(x.left);
    foo(x.right);
}
```



17 11 4 2 10 13 16 14 20 18 25 24

(b) Assume the delete() method provided by the BST API is implemented based on the deletion discussed in class. Given the BST in (a), a client program calls the delete() method to remove the node with key "20", and then calls the delete() method again to remove the node with key "13".

a. **(10 points)** Draw the resulting tree after removing the 2 nodes.



b. **(5 points)** Write the output when the client program calls the print() method after removing the 2 nodes. 17 11 4 2 10 14 16 24 18 25

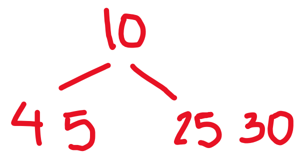
Name: ____ NetID: ____________________________

Problem 3 – 2-3 Trees (30 points)

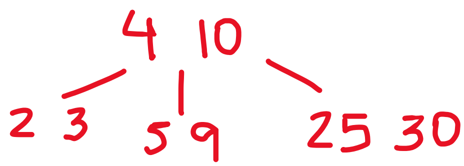
Construct a 2-3 tree whose keys are inserted in the following sequence.

5 25 10 4 30 2 9 3 28

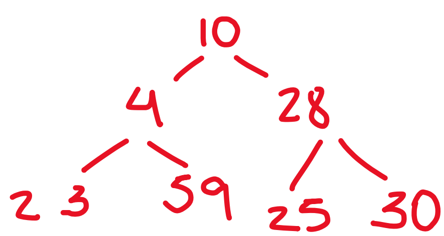
(a) **(6 points)** Show the 2-3 tree after the insertions of 5 25 10 4 30



(b) **(6 points)** Show the 2-3 tree after the insertions of 5 25 10 4 30 2 9 3



(c) **(6 points)** Show the 2-3 tree after the insertion sequence completed.



(d) **(3 points)** What is the number of links (perfect balance) of the tree from part (c)? 2

(e) **(6 points)** Briefly explain why we would use a 2-3 tree over a standard BST?

[3 points] BST worst case for search/insert/delete is O(n), the tree is skewed (not balanced)
[3 points] 2-3 trees worst case for search/insert/delete is O(log n), tree is always balanced.

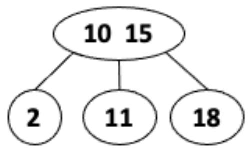
More in depth explanation:
In BST, the search, insertion and delete are O(n) in the worst case, as the insertion sequence determines the shape of the tree and doesn't guarantee a balanced tree.

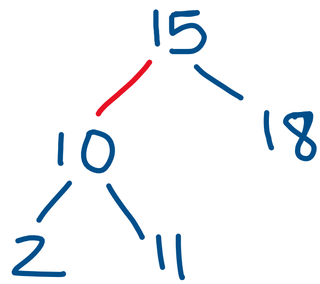
Name: ____ NetID: ____________________________

To guarantee O(log n) for search and insertion, 2-3 Tree is perfectly balanced by allowing a node to hold more than one key, 2-node, or 3-node. The worst case is all 2-nodes $O(\log_2 n)$, and the best case is all 3-nodes, thus, $O(\log_3 n)$

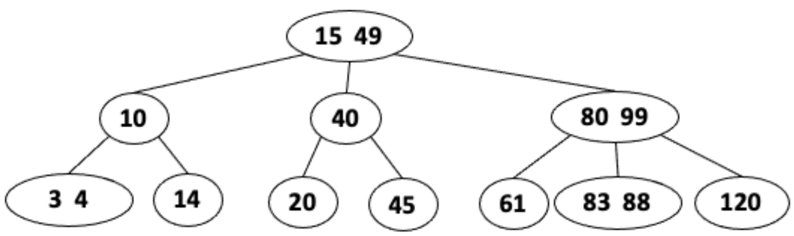
Problem 4 – Left-Leaning Red-Black Tree (42 points)

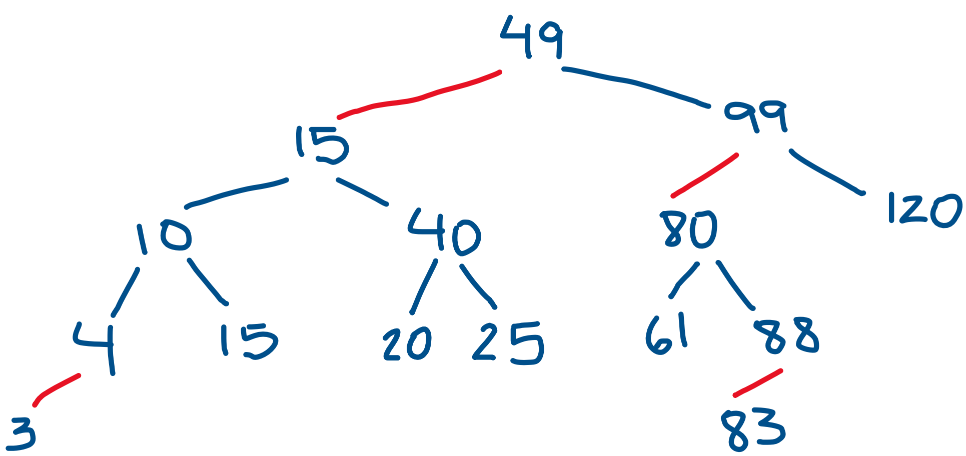
a) **(5 points)** Draw the corresponding left-leaning red-black tree to the 2-3 tree below. Label red links R.





b) **(15 points)** Draw the corresponding left-leaning red-black tree to the 2-3 tree below. Label red links R.





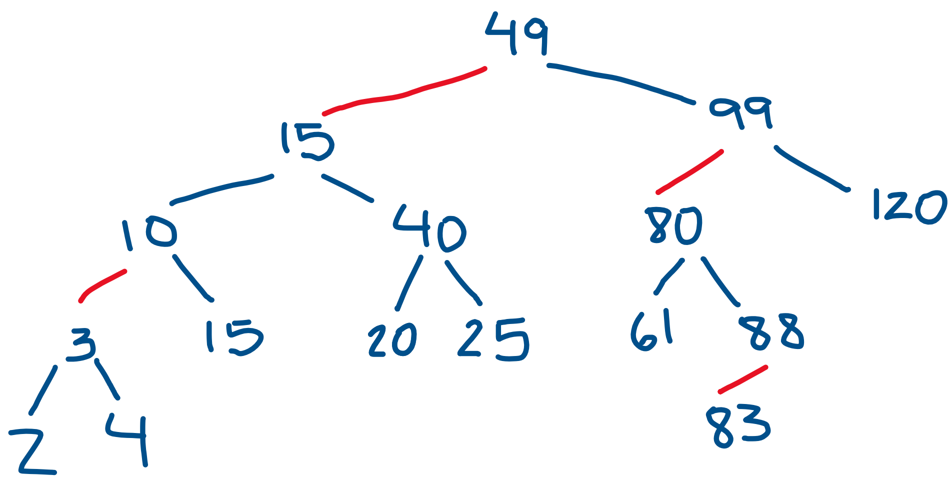
c) **(5 points)** What is the number of black links (perfect black balance) in the left-leaning red-black tree from (b) in any of the paths from the root to null links. 2

Name: ______________________________________ NetID: ___________________________

d) **(7 points)** What is the minimum height (tilde notation) of a LLRB tree in terms of *n*, where *n* is the number of items in the tree? Why?
~log n // 3 points
The tree has the minimum height when there are only 2-nodes in the tree. // 4 points

Name: ______________________________ NetID: ______________________

e) (10 points) Assume the insert() method provided by the LLRB API is implemented based on the insertion discussed in class. Given the LLRB tree in (b), a client program calls the insert() method to insert the node with key "2". Draw the resulting tree after the node is inserted.



Name: ______________________________________ NetID: ___________________________

Problem 5 – Priority Queue (30 points)

a) **(5 points)** Is an array that is sorted in increasing order a MIN binary heap? Why?
Yes. An array in increasing order has the heap shape (it is a complete binary tree) property and it also has the heap order property (parent node key <= children's keys).

b) Below is an array representing a valid binary heap for a MIN heap.

index

Key[]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		2	3	9	7	5	12	20	7	11	20	14	18			

(i) **(12 points)** Show the array contents after 2 insertions: first insert key **6** and then insert key **1**.

// 2 points for each correct 1 (index 1), 2 (index 3), 9 (index 6), 6 (index 7), 12 (index 13), 20 (index 14)

// 2 points for all other keys

index

Key[]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		1	3	2	7	5	9	6	7	11	20	14	18	12	20	

(ii) **(13 points)** After the above insert operations, assume 2 (two) delMin() operations were performed, show the contents of the array.

// 2 point for each correct 3 (index 1), 5 (index 2), 6 (index 3), 12 (index 5), 20 (index 6)

// 1 points for each indices 13 and 14 empty

// 2 points for all other keys

index

Key[]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		3	5	6	7	12	9	20	7	11	20	14	18			