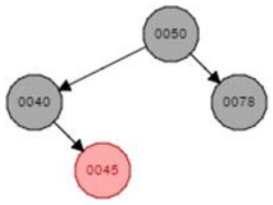
CS 385, Homework 5: Balanced Trees and Transform-and-Conquer

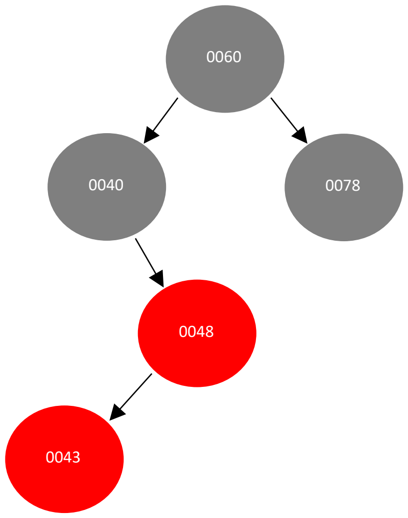
Name: Natalie Zoladkiewicz   Date: 11/26/2023

Points earned: ____ / 74 = ____ %

1. Show the red-black tree after inserting a node with the key 0043. Use the document on Canvas that explains the insertion process succinctly. List the case you applied (i.e. 1, 2a, 3b), and write the steps you took to fix the tree (also listed in the document).



a) Draw the tree after doing a regular binary search tree insertion of 0043. (3 points)

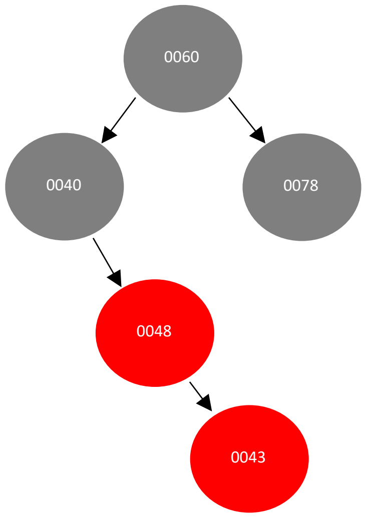


b) Which RBT property is violated? (3 points) The property that states that a red parent cannot have a red child is violated.

Case seen after the regular binary search tree insertion: (3 points) There is a red node has a red left child node.

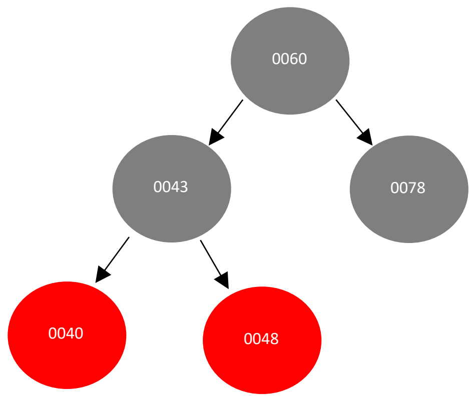
Steps taken to fix the tree: (3 points) We can make a right rotation.

Draw the tree after taking the steps you just described. (3 points)



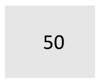
c) Which property is violated now? (3 points) We still have the same property being violated. The same red node has a red child.

d) Case seen after first fixup: (3 points) The red node 0048 has a red right child 0043.
Steps taken to fix the tree: (3 points) We can do a left rotation.
Draw the tree after taking the steps you just described. (3 points)

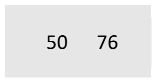


2. Draw the 2-3 tree after inserting each of the following keys. Redraw the whole tree for each part.

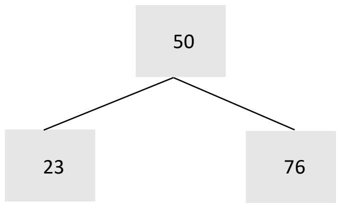
a) 50 (1 point)



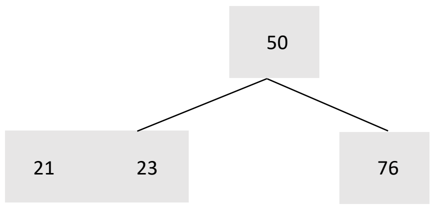
b) 76 (1 point)



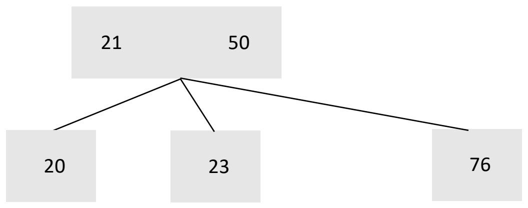
c) 23 (3 points)



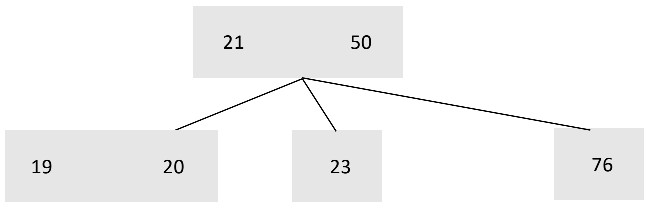
d) 21 (3 points)



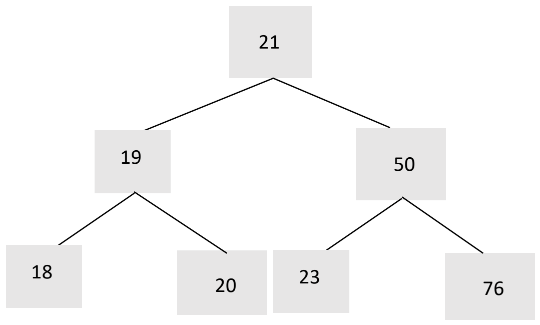
e) 20 (3 points)



f) 19 (3 points)



g) 18 (3 points)



3. Read pages 241-242 in the textbook. Using that information, write pseudocode for computing the LCM of an array A[1..n] of integers. You may assume there is a gcd() function available. (6 points)

```
ALGORITHM LCM(A[1..n]):
   // Computes the least common multiple of all the integer in array A
   if n == 1:
         return A[1]
   result = A[1]
   for i = 1 to n:
         gcd_result = gcd(result, A[i])
         result = (result * A[i]) / gcd_result
   return result
```

4. Horner's method: $p(x) = 4x^4 + 5x^3 - 2x^2 - 4x + 7$
   a) Repeatedly factor out x in the polynomial above so that you can apply Horner's method. Write your final expression for $p(\mathrm{x})$. (5 points)

   $$p(x) = \big(((4x + 5)x - 2)x - 4\big)x + 7$$

   b) Show values of the array P[0..n] as needed to apply Horner's method. (3 points)

   [4, 5, -2, -4, 7]

   c) Apply Horner's method to evaluate the polynomial at x 2. Make a table as we did in class showing the values x, p, n, and i, and then state your final answer for $p(2)$. (5 points)

x	p	n	i
2	4	4	1
2	5	13	2
2	-2	24	3
2	-4	44	4
2	7	95	5

   $p(2)$ = 95

   d) Use **synthetic** (not long) **division** to divide $p(x)$ by x - 2 to check your work. Be sure to show your work. (5 points)
   $p(x) = 4x^4 + 5x^3 - 2x^2 - 4x + 7$

$x - 2 = 0$	4	5	-2	-4	7
$x = 2$	0	8	26	48	88
	4	13	24	44	95

5. Rewrite the *LeftRightBinaryExponentiation* algorithm on page 237 in the textbook to work for n = 0 as well as any positive integer. *No credit will be given for answers that simply start with an if statement for n = 0.* (6 points)

```
ALGORITHM LeftRightBinaryExponentiation(a, b(n)):
    // Computes a^n
    product ← 1
    for i ← 1 down to 0 do
        product ← product * product
        if b_1 = 1 product ← product * a
    return product
```