

Name: Marinna Ricketts-Uy

Section: 02 – Budhraj



HW #: 5

Version: A

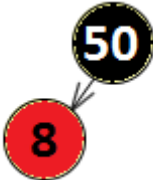
Username: pd12778

1. Inserting into Red-Black Tree: 50 8 20 15 100 150 18 30 40

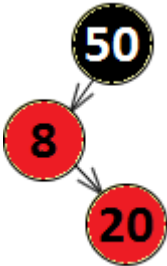
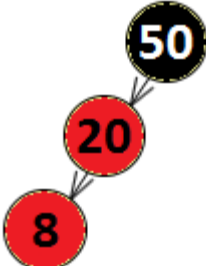
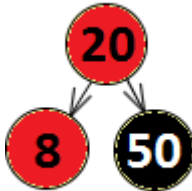
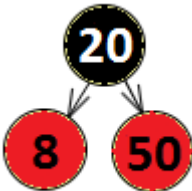
Inserting 50:

Step 1: Insert 50 as red 	Step 2: Change the node to black 	Since there is no other nodes in the tree, 50 is inserted as red and then simply changed to black. It is changed to black because the root must be black.
--	--	---

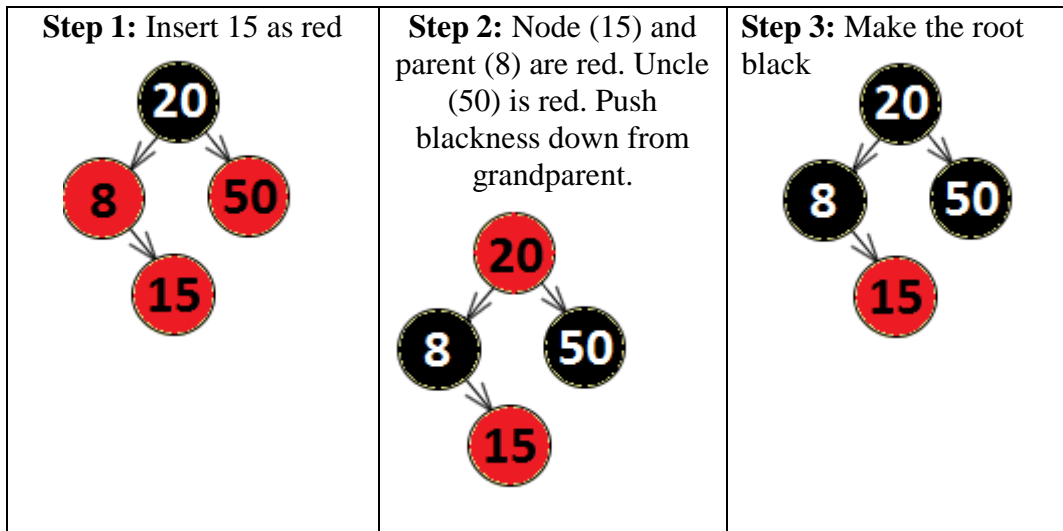
Inserting 8:

Step 1: Insert 8 as red 	Step 2: We change nothing and just make sure the root is black
---	---

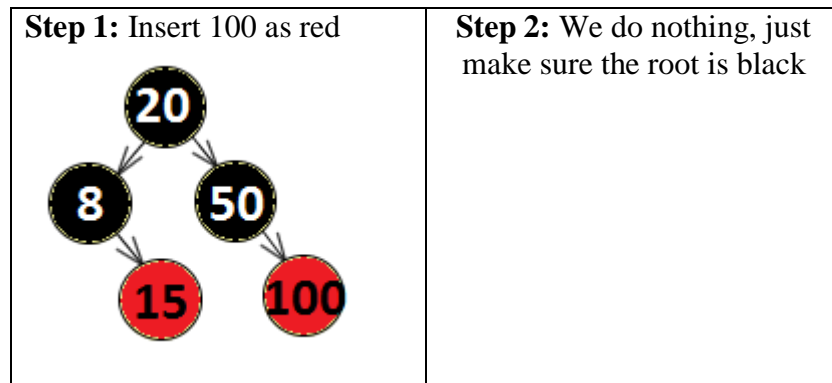
Inserting 20:

Step 1: Insert 20 as red 	Step 2: Node (20) and parent (8) are red. Node is a left child, parent is a right child. Rotate left. 	Step 3: Node (8) and parent (20) are red. Node is left child, parent is left child. Rotate right. 	Step 4: Node (20) and Node (50) swap colors. 
--	---	--	--

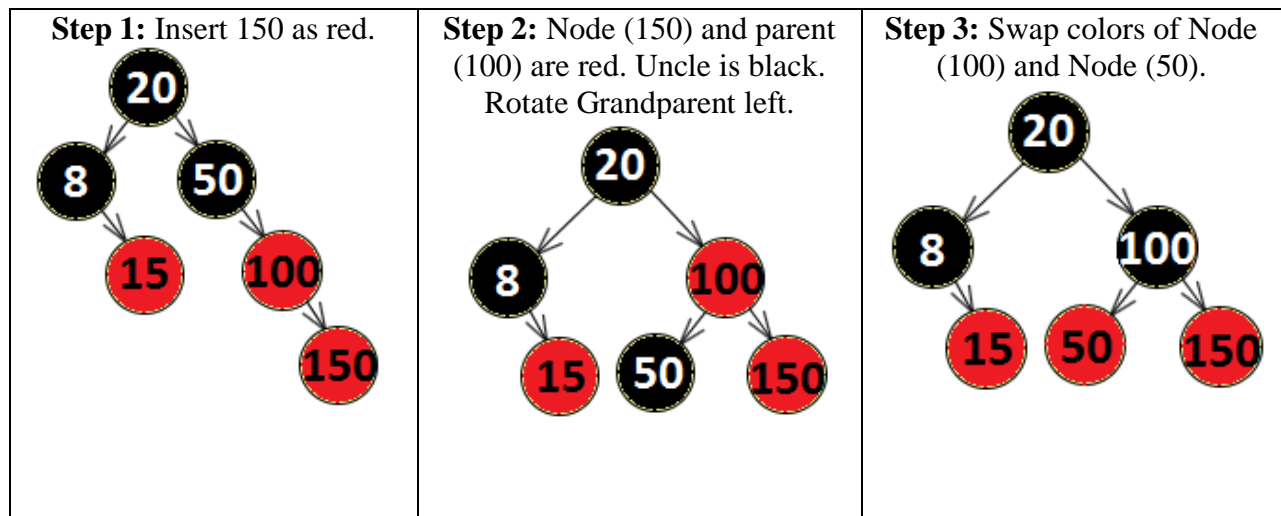
Inserting 15:



Inserting 100:

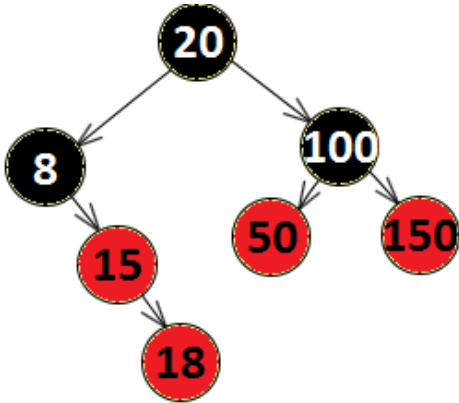


Inserting 150:

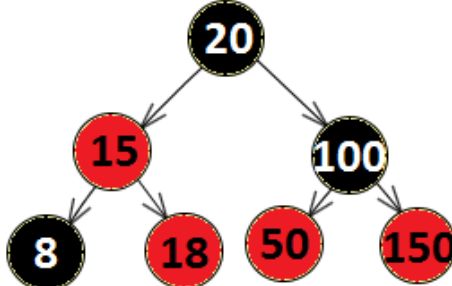


Inserting 18:

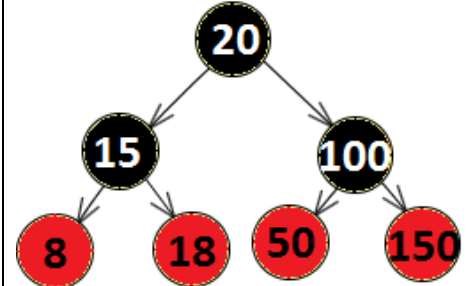
Step 1: Insert 18 as red.



Step 2: Node (18) and parent (15) are red. Uncle is black. Rotate left on grandparent (8).

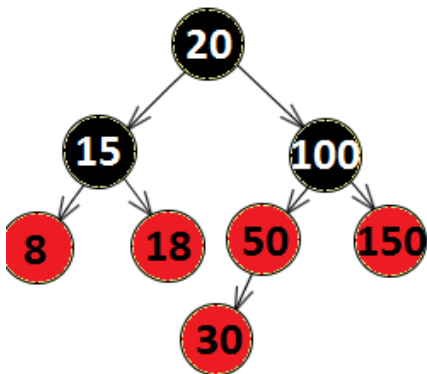


Step 3: Swap colors of Node (15) and Node (8).

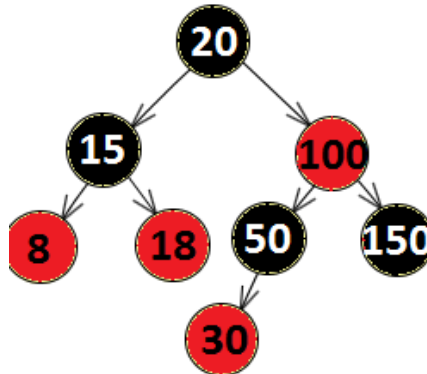


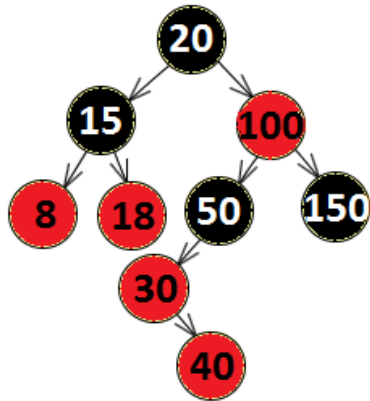
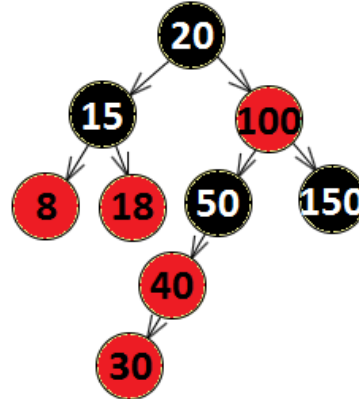
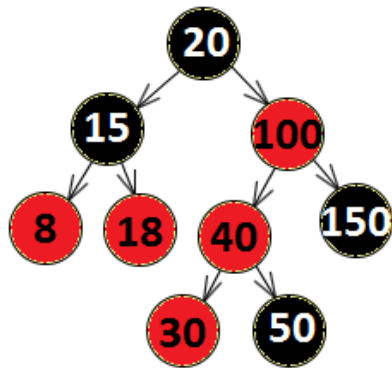
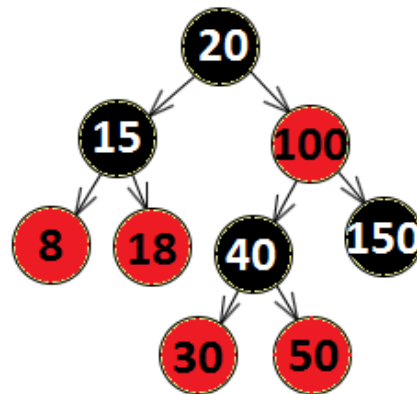
Inserting 30:

Step 1: Insert 30 as red.

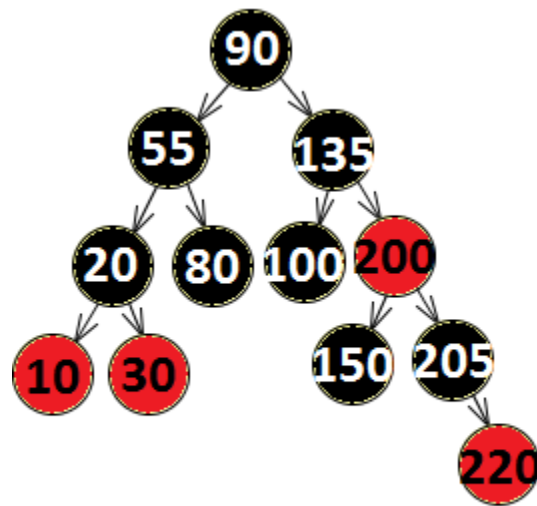


Step 2: Node (30) and parent (50) are red. Uncle (150) is red. Push blackness down from grandparent.

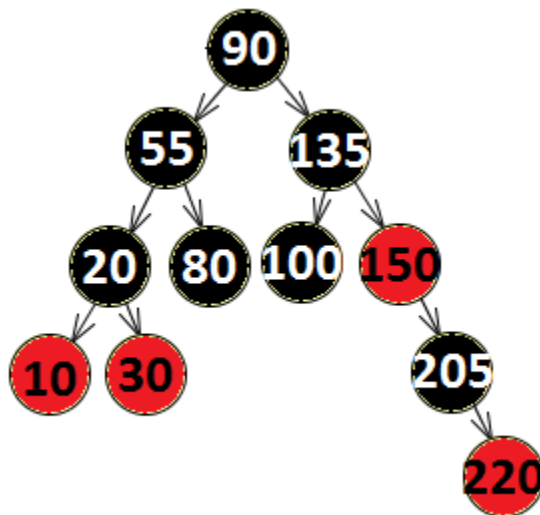


Inserting 40:**Step 1:** Insert 40 as red.**Step 2:** Node (40) and parent (30) are red. Node is a right child, parent is a left child. Rotate left.**Step 3:** Node (30) and parent (40) are red. Node is left child, parent is left child. Rotate right**Step 4:** Swap color of Node (40) and Node (50).

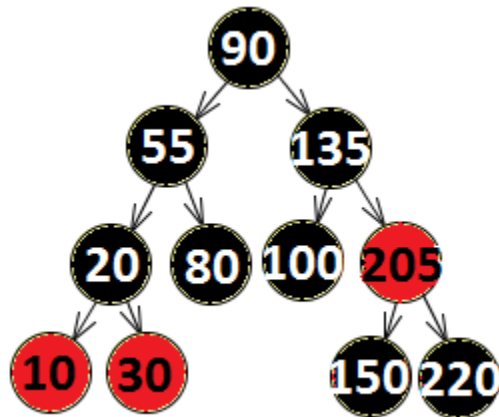
2.

Original Red-Black Tree:**Delete 200:**

The node to delete has two children. In order to delete it, first find the largest node in the left subtree. In this case, the largest is 150. Copy largest value of left subtree into the node to delete. And remove the node whose value was copied. This step is shown below

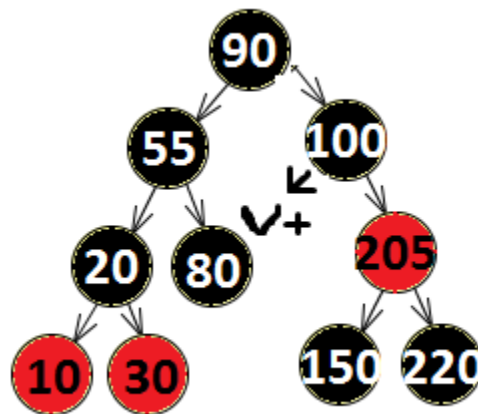


Rotate left on 150

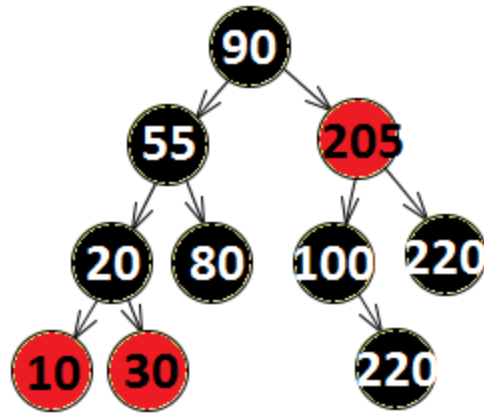


Delete 135:

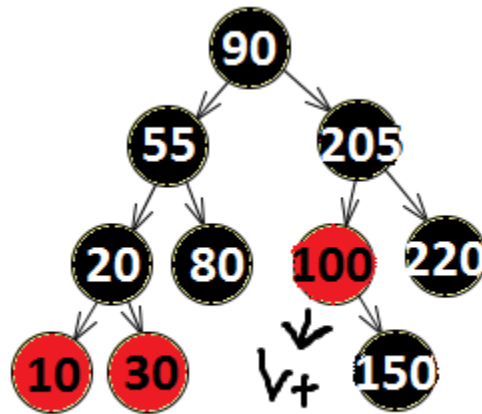
Node to delete has two children. In order to delete 135, find the largest node in the left subtree. Then copy the largest value into the node to delete. Lastly, remove the node whose value was copied. This step is shown below:



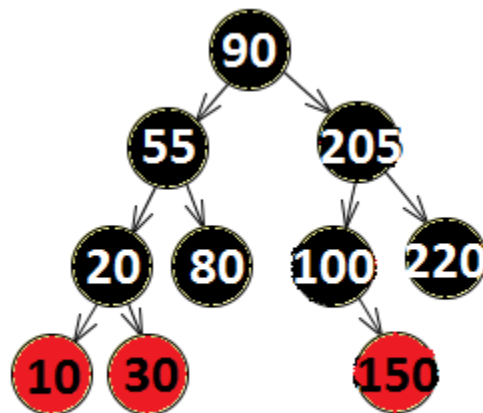
Now we need to do rotations. V+ has a red sibling, and the sibling's children are black. This resembles Case 1 for Bottom up Deletion. In order to fix this, first rotate the sibling around the parent:



Then we swap colors of the sibling and the parent:



Still have to fix the extra “blackness”. The parent is red and the sibling is black, so this falls under Case 2. The colors of the parent and the sibling are swapped:



3. Bottom up Insertion:

Describe when you have to use the zig-zag method while inserting a node into a red black tree. How does it work?

- For Case 2 and Case 3, you have to use the zig-zag method. In Case 2, it occurs when the Parent is red, but the Uncle is black and X and its parent are opposite type children. You need to color Grandparent red, color X black, rotate left(or right) on parent, and rotate right(or left) on Grandparent. In Case 3, it occurs when the Parent is red, but the Uncle is black and X and its parent are both left (right) children. You need to color the Parent black, color Grandparent red, and rotate right (left) on Grandparent.

4. What is the height of a red-black tree with n nodes? Explain why your answer is correct.

- The height of a red-black tree with n nodes is $O(\log n)$. The reason behind this is that the black depth of the red-black tree is $O(\log n)$, where n is the number of black nodes. When you consider the height of the red-black tree, you only care about the black nodes. If you remove all the red nodes, the black depth of the leaves is the same as the height of the tree. This means we have a complete tree, since all leaves have the same exact depth. Then if the black depth of the original tree is equal to the height of the tree with no red nodes, and the height of the tree with no red nodes is $O(\log n)$, then the black depth of the original tree is $O(\log n)$. So this means the height of a red-black tree with n nodes is $O(\log n)$.

5.

Inserting keys: **15 76 72 77 40 47 69 94 68 14 35 32 80 92 70 2 60 61 99 59 45 8 29 86**
30 93 88 42 78

Linear Probing:

$$h(k) = k \bmod 29$$

0	29	29 mod 29 = 0 (1 probe)
1	59	59 mod 29 = 1 (1 probe)
2	2	2 mod 29 = 2 (1 probe)
3	32	32 mod 29 = 3 (1 probe)
4	60	60 mod 29 = 2 (3 probes)
5	92	92 mod 29 = 5 (1 probe)
6	35	35 mod 29 = 6 (1 probe)
7	94	94 mod 29 = 7 (1 probe)
8	61	61 mod 29 = 3 (6 probes)
9	8	8 mod 29 = 8 (2 probes)
10	68	68 mod 29 = 10 (1 probe)
11	40	40 mod 29 = 11 (1 probe)
12	69	69 mod 29 = 11 (2 probes)
13	70	70 mod 29 = 12 (2 probes)
14	72	72 mod 29 = 18 (1 probe)
15	15	15 mod 29 = 15 (1 probe)
16	14	14 mod 29 = 14 (3 probes)
17	99	99 mod 29 = 12 (6 probes)
18	76	76 mod 29 = 18 (1 probe)
19	77	77 mod 29 = 18 (1 probe)
20	47	47 mod 29 = 18 (3 probes)
21	45	45 mod 29 = 16 (6 probes)
22	80	80 mod 29 = 22 (1 probe)
23	30	30 mod 29 = 1 (23 probes)
24	93	93 mod 29 = 6 (18 probes)
25	88	88 mod 29 = 1 (25 probes)
26	42	42 mod 29 = 13 (14 probes)
27	78	78 mod 29 = 20 (8 probes)
28	86	86 mod 29 = 28 (1 probe)

	# of Probes
Linear	136

Quadratic Probing:

$$h(k) = k \bmod 29$$

$$N = 29$$

0	29
1	59
2	2
3	32
4	61
5	92
6	35
7	94
8	8
9	42
10	68
11	40
12	69
13	70
14	72
15	15
16	99
17	45
18	76
19	77
20	
21	30
22	47
23	14
24	88
25	78
26	80
27	60
28	86

93 is undetermined.

	# of Probes
Quadratic	Undetermined