4DS3: Data Structures and Algorithms Spring/Summer 2023
Challenge Project 3

1. We have a linear probing table containing 15 slots, numbered 0, 1, 2, …, and 14. For the sake of simplicity, we'll assume that we're hashing integers and that our hash function works by taking the input integer and returning its last digit. (This is a terrible hash function, by the way, and no one would actually do this).
a. Draw the linear probing table formed by inserting 31, 41, 59, 26, 53, 58, 97, 133, and 93, in that order, into an initially empty table with ten slots.

Linear Probing
31 mod 15=1
41 mod 15=11
59 mod 15=14
26 mod 15=11⇒ (11+1) mod 15=12
53 mod 15=8
58 mod 15=13
97 mod 15=7
133 mod 15=13⇒ (13+1) mod15=14 ⇒ (14+1) mod 15 =0
93 mod15=3

Develop initial hash table
Initial hash:
key[1, 11, 14, 11, 8, 13, 7, 13, 3]
data{1: [31], 11: [41, 26], 14: [59], 8: [53], 13: [58, 133], 7: [97], 3: [93]}
Linear Probing:
Key[1, 11, 14, 12, 8, 13, 7, 0, 3]
data:{1: [31], 11: [41], 14: [59], 12:[26], 8: [53], 13: [58], 7: [97], 0:[133], 3: [93]}

0	**133**
1	**31**
2	
3	**93**
4	
5	
6	

7	**97**
8	**53**
9	
10	
11	**41**
12	**26**
13	**58**
14	**59**

b. What is the load factor of the table?

Load factor=slot_size/number of keys=9/15=0.6=60%

c. Draw a different linear probing table that could be formed by inserting the same elements given above into an empty, ten-slot table in a different order than the one given above, or tell us that it's not possible to do this. Assume that you're using the same hash function.

Initial Hash: slot_size =10
key[1, 1, 9, 6, 3, 8, 7, 3, 3]
data{1: [31, 41], 9: [59], 6: [26], 3: [53, 133, 93], 8: [58], 7: [97]}

31 mod 10=1
41 mod 10=1⇒(1+1) mod 10=2
59 mod 10=9
26 mod 10=6
53 mod 10=3
58 mod 10=8
97 mod 10=7
133 mod 10=3⇒ (3+1) mod10=4

93 mod10=3⇒10+1 mod 10=1⇒1+1 mod 10=2 ⇒2+1 mod 10 =3⇒3+1 mod 10=4⇒ 4+1 mod 10=5

0	
1	**31**
2	**41**
3	**53**
4	**93**
5	**133**
6	**26**
7	**97**
8	**58**
9	**59**

Load factor for this table is 9/10=90%

d. Which slots do you have to look at to see if the table contains the number 72? Assume the table contains 10 slots:
Division method:72 mod 10=2
Multiplication:m=10,A=17/32
m(kA mod 1)=10 x (72 x17/32 mod 1)=2.5≈3

e. Which slots do you have to look at to see if the table contains the number 137?
Division method: 137 mod 10=7
Multiplication method:m=10,A=17/32
M (kA mod 1)=10 x (137 x 17/32) mod 1=7.8 ≈8

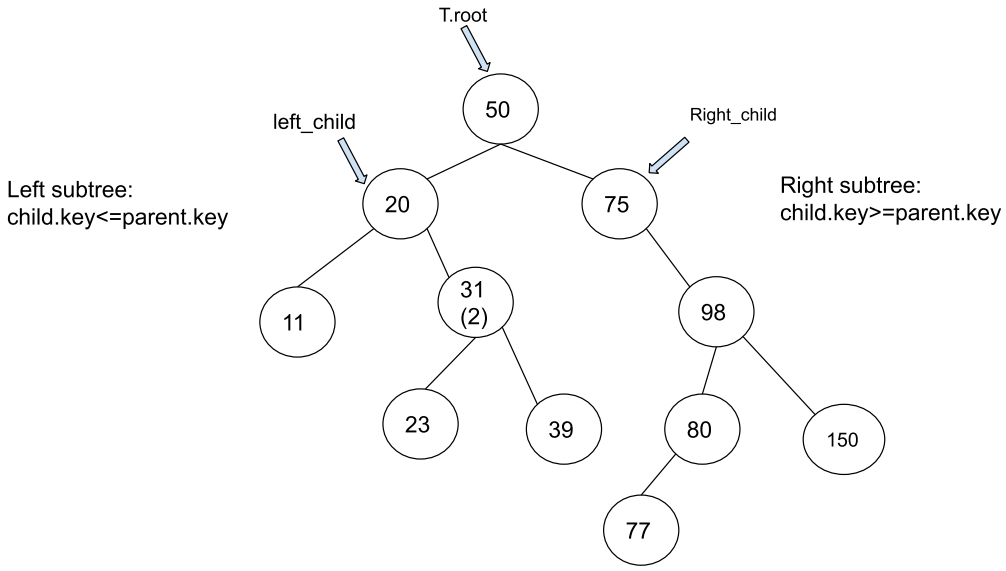
f. Draw the table formed by starting with the table in part a) and the inserting the elements 106, 107, and 110, in that order.
Using linear probing to handle collision:

106 mod 15 =1 ⇒ (1+1) mod 15 =2
107 mod 15 = 2 ⇒ (2+1) mod 15 =3 ⇒ (3+1) mod 15 =4

110 mod 15 = 5

0	**133**
1	**31**
2	**106**
3	**93**
4	**107**
5	**110**
6	
7	**97**
8	**53**
9	
10	
11	**41**
12	**26**
13	**58**
14	**59**

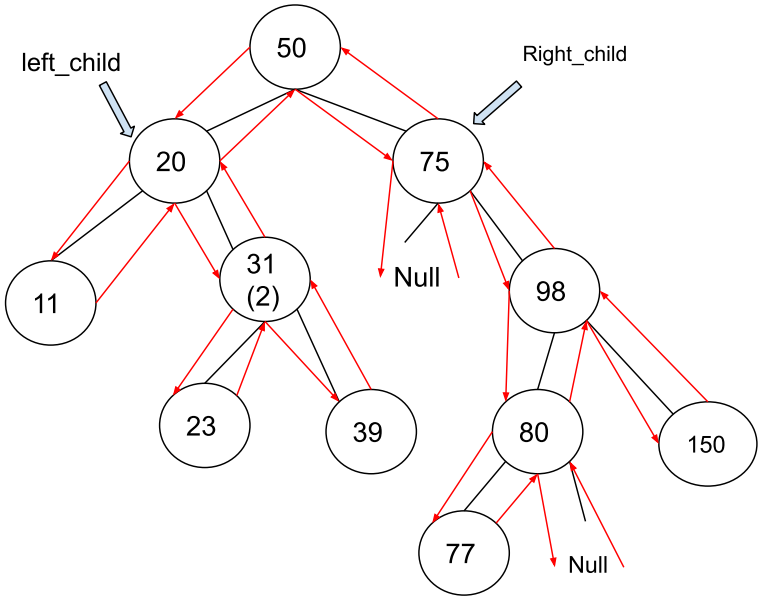
2. Draw what a binary search tree would look like if the following values were added to an initially empty tree in the following order. If needed, explain what logic was used to insert the node.
50, 20, 75, 98, 80, 31, 31, 150, 39, 23 11, 77



{50,20,75,98,80,31,31,150,39,23,11,77}

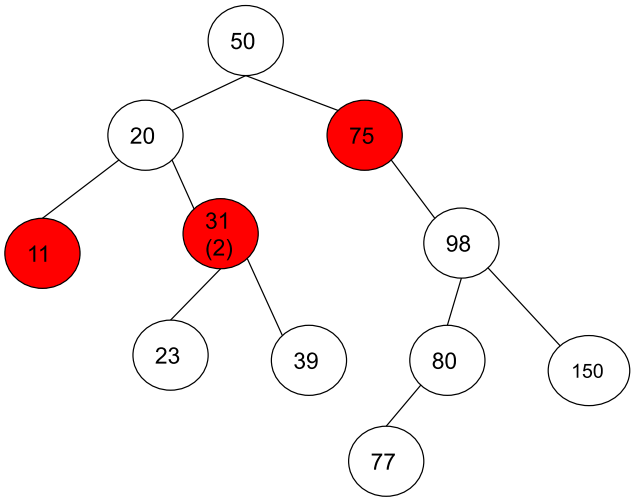
a. What would be the output of an in-order walk on this tree?

Inorder Tree walk:

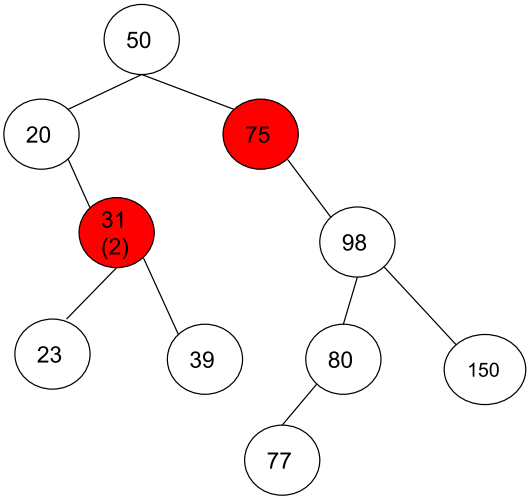


{11,20,23,31,31,39,50,75,77,80,98,150}

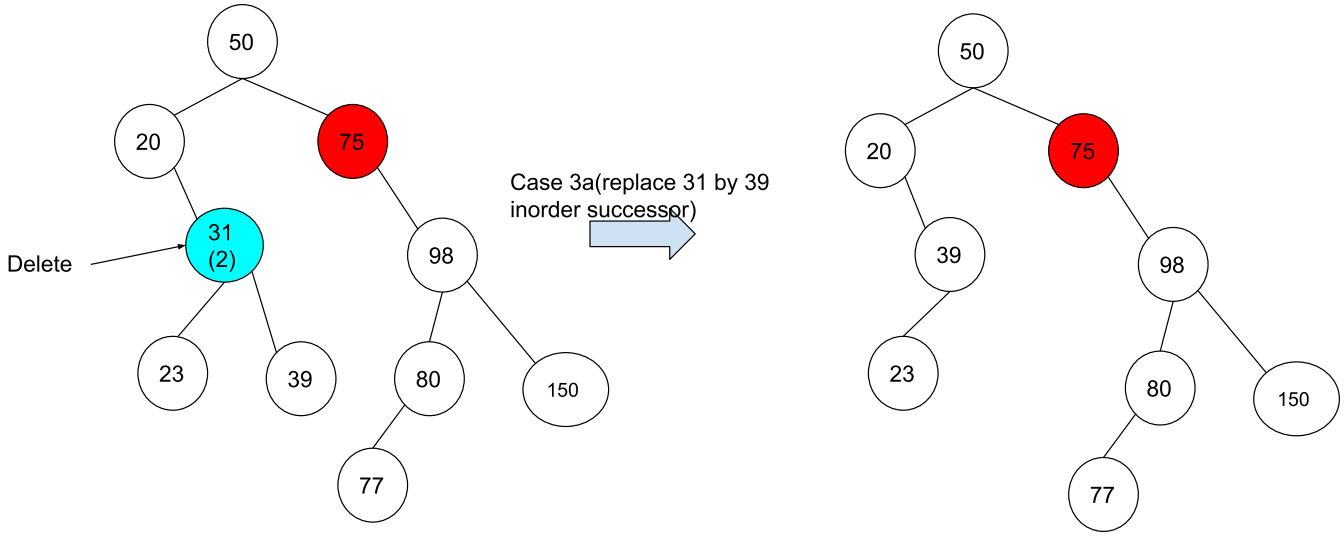
b. Show what the tree will look like if you delete the following nodes. In each case, start with the original tree.
11, 31, 75



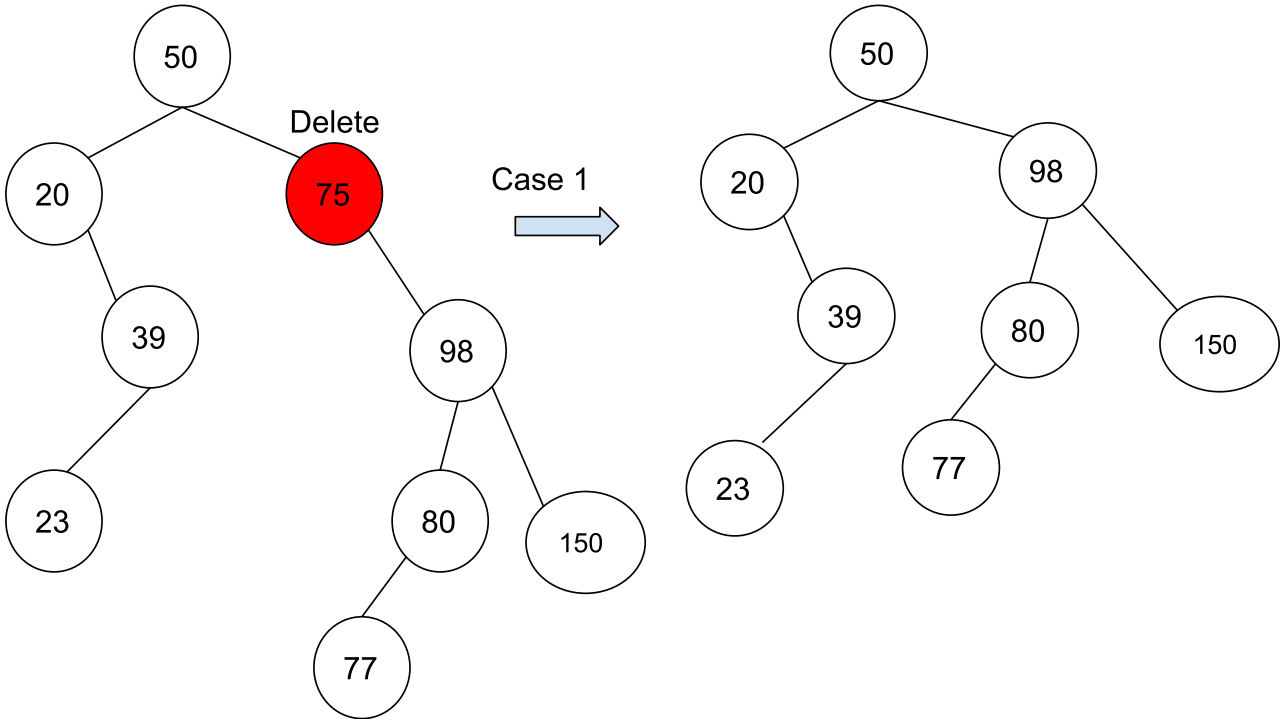
Delete 11:



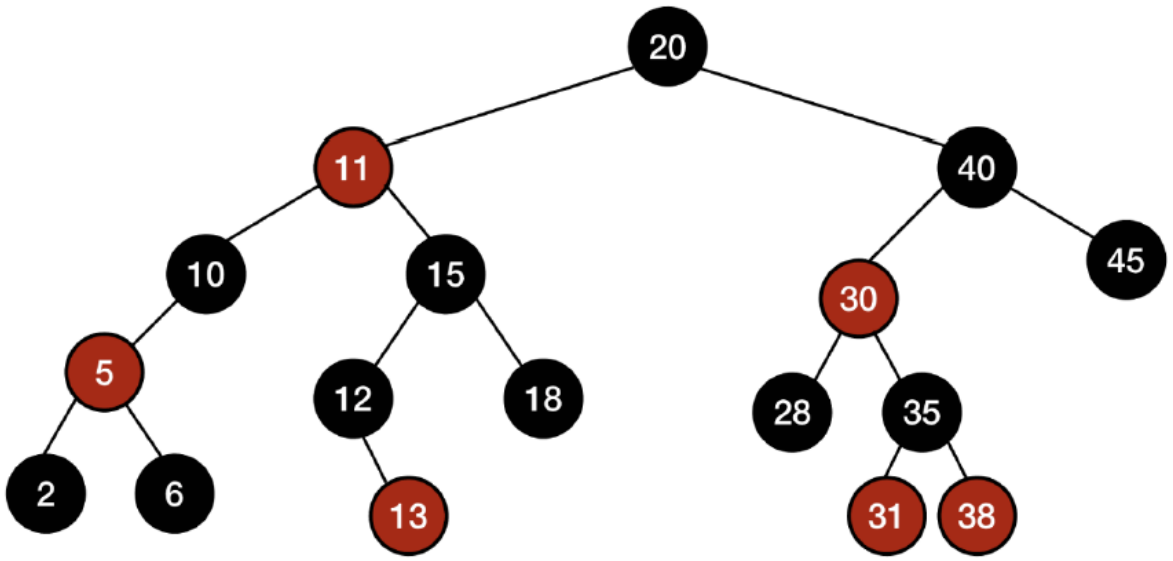
Delete31:



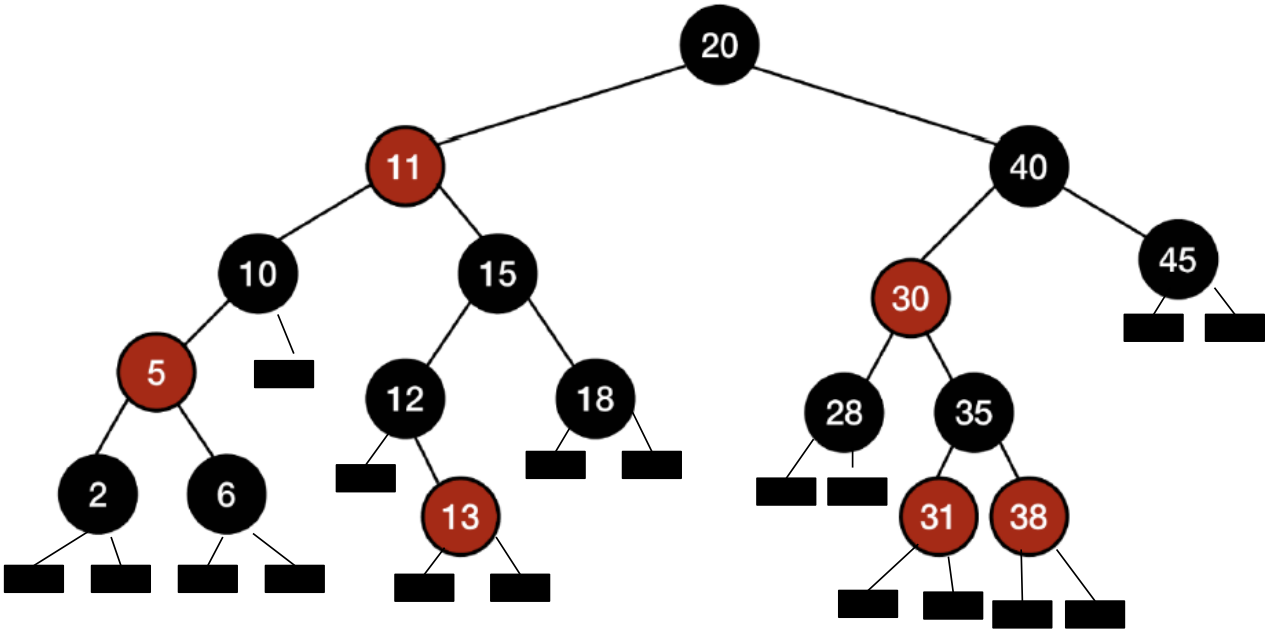
Delete 75:



3. Consider the following tree:



Is this a valid Red-Black Tree? Why or why not? Justify your answer in full.



1. **Nodes are either red or black.√**
2. **The root and the NIL are black√**
3. **If a node is red, then its children are black√**
4. **All paths from the node to NIL descendants have the same number of black nodes:X**

   Node 20:

   Path 1:20 ->11->10->5->2-> NIL x2

   XPath 2:20 ->11->10->NIL ⇐ Not the same number black nodes from node 20 to NIL as others

   Path 3:20 ->11->10->5->6-> NIL x2

Path 4:20 ->11->15->12->NIL
Path 5:20 ->11->15->12->13->NIL x2
Path 6:20 ->11->15->18->NILx2
Path 7:20 ->40->15->NILx2
Path 8:20 ->40->30->28->NIL x2
Path 9:20 ->40->30->35->31->NIL x2
Path10:20 ->40->30->35->38->NIL x2