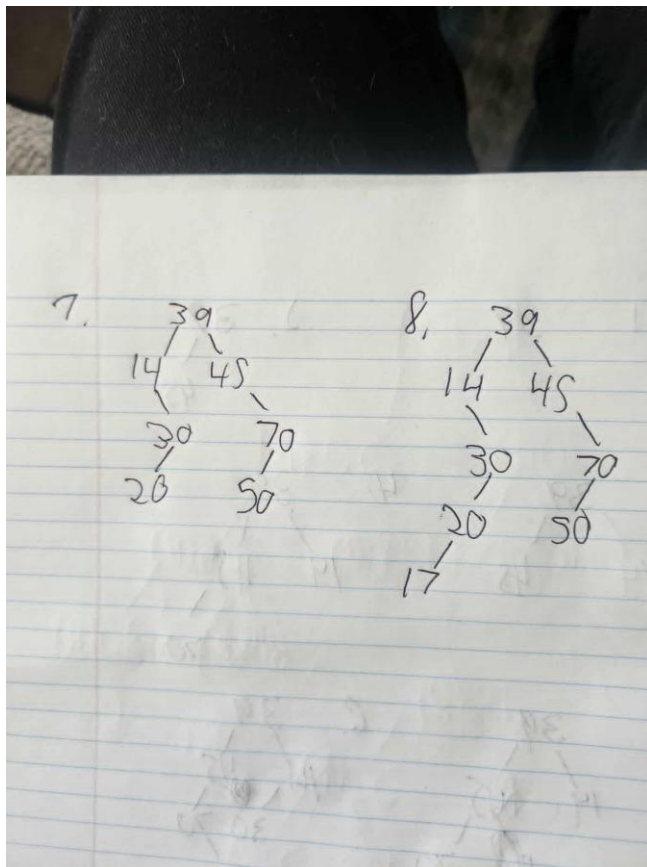
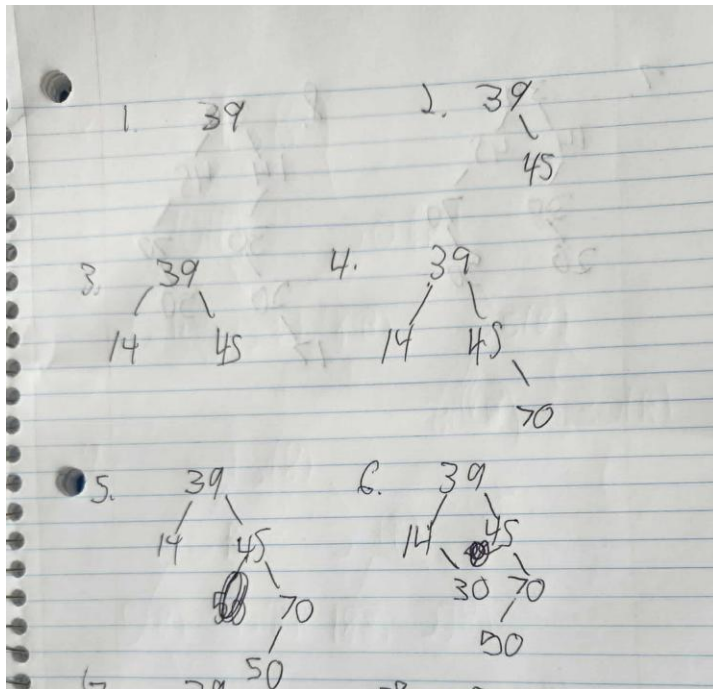


Part 1

Problem 1:

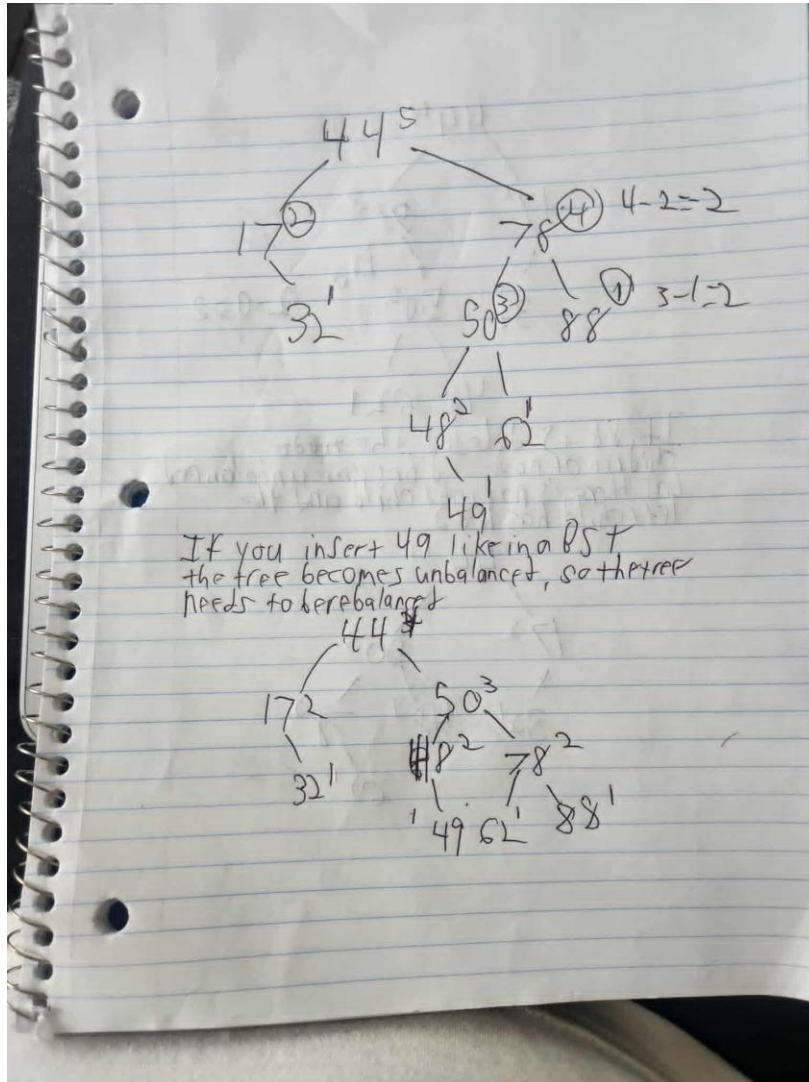


Part 2

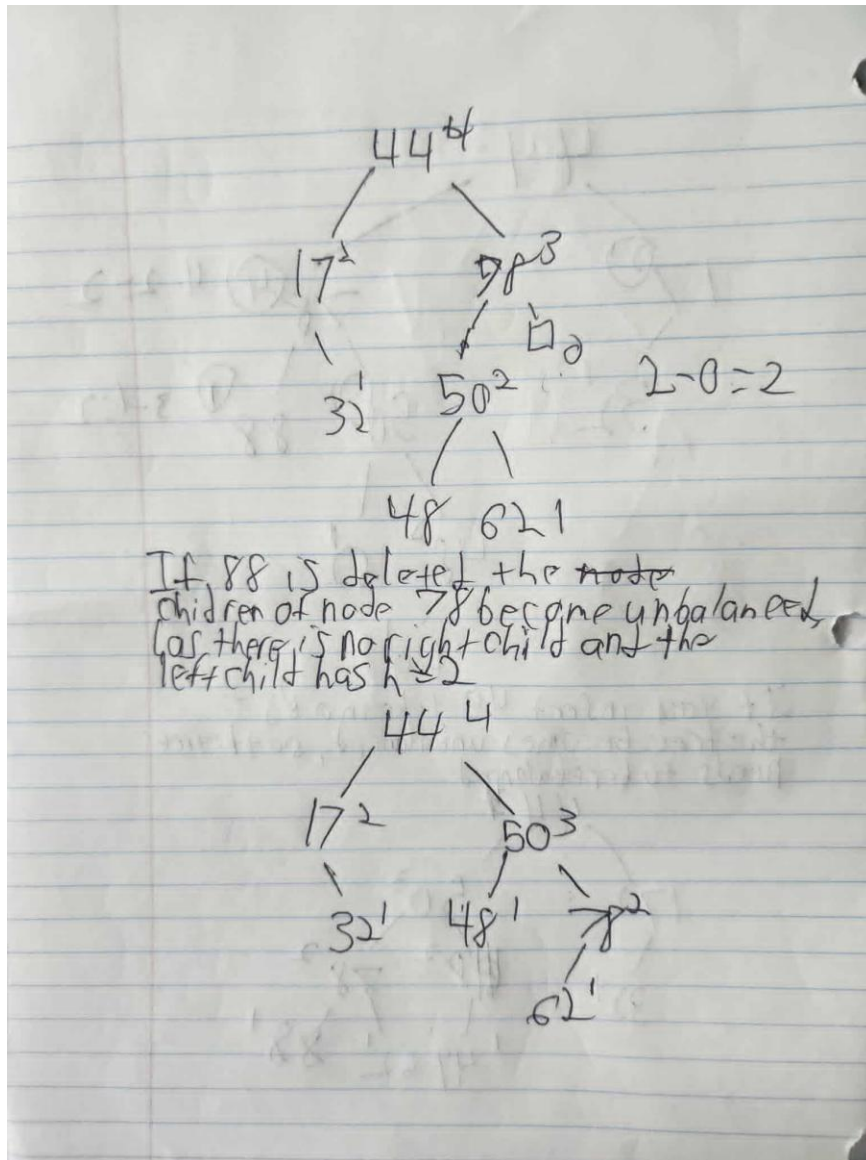
Problem 3:

A. AVL Trees:

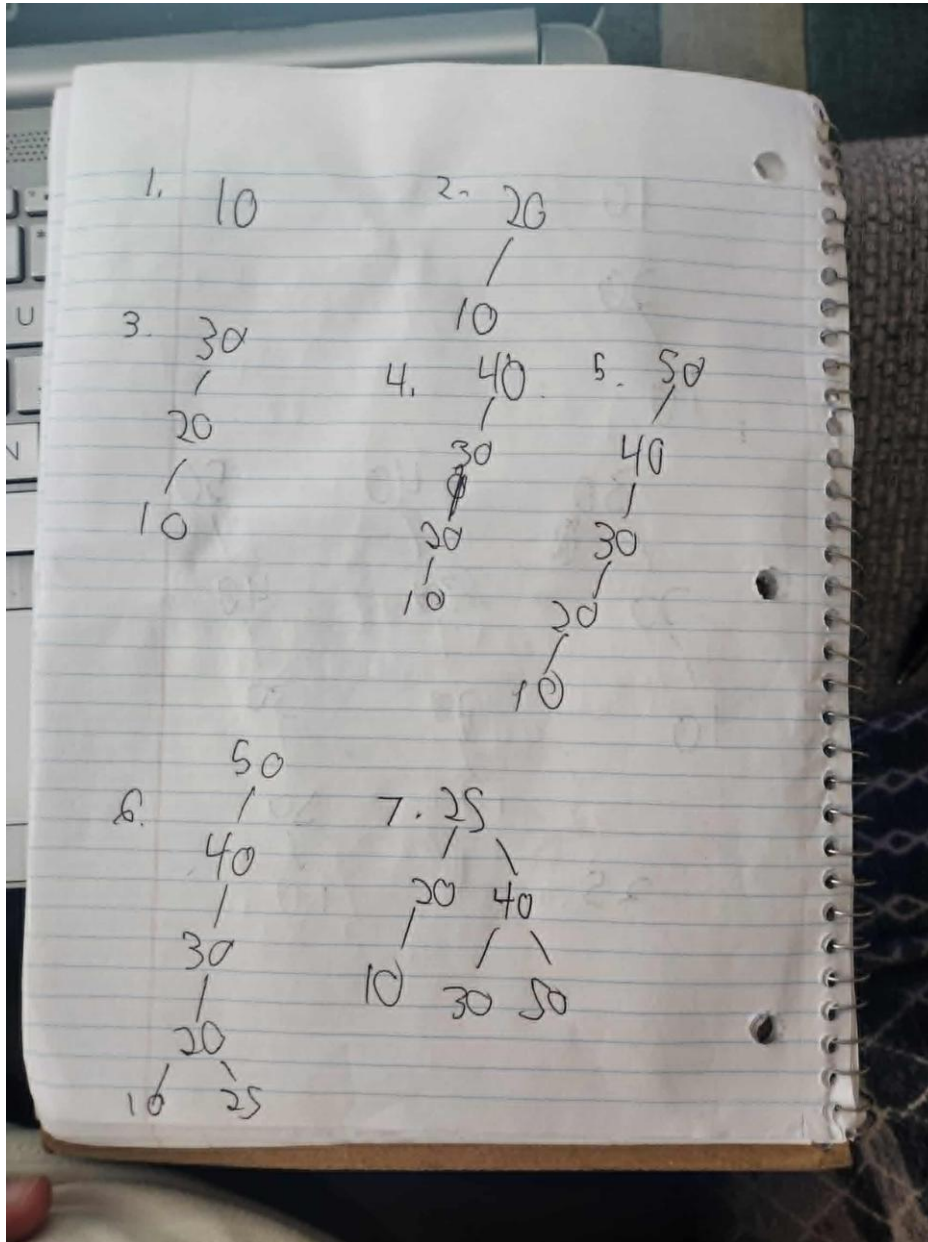
Task 1:



Task 2:



B. Splay Trees



Since 25 was the last key inserted, it is already at the root and the tree doesn't have to be reorganized when it is searched for. If you were to search for another key however, the tree would be reorganized around that key being the root.

C. 2,4 Trees

1. 5
2. (5, 10)
3. (5, 10, 15)

4. $\begin{array}{c} 10 \\ / \quad \backslash \\ 5 \quad (15, 20) \end{array}$ Split, make middle value parent.

5. $\begin{array}{c} 10 \\ / \quad \backslash \\ 5 \quad (15, 20, 25) \end{array}$

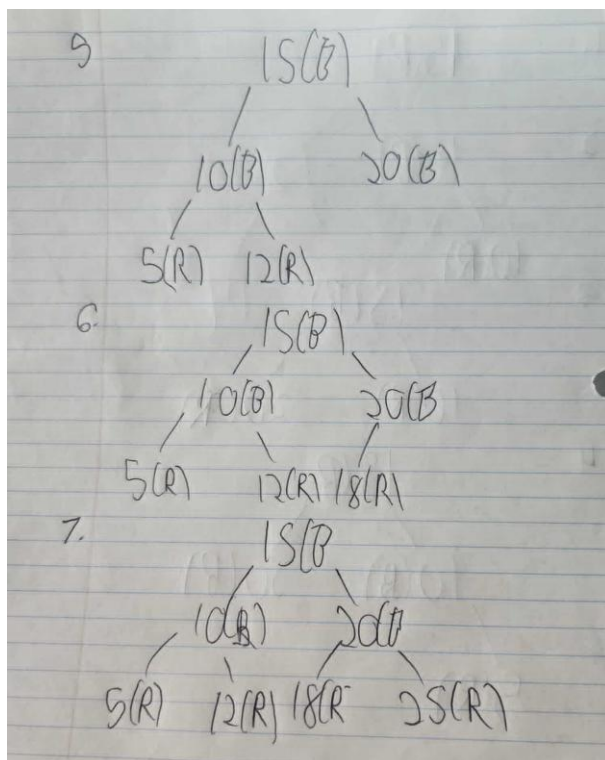
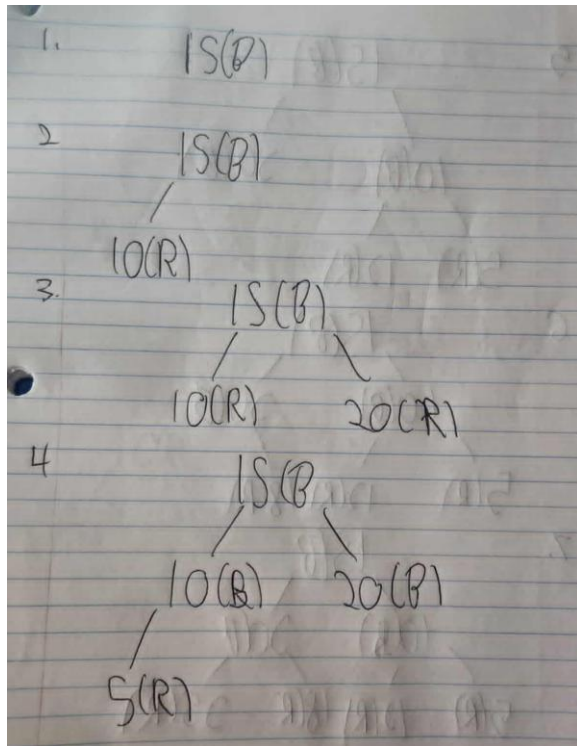
6. $\begin{array}{c} (10, 20) \\ / \quad \backslash \\ 5 \quad 15 \quad (25, 30) \end{array}$ Split right child. move 20 to parent and 15 to middle child

7. $\begin{array}{c} (10, 20) \\ / \quad \backslash \\ 5 \quad 15 \quad (25, 30, 35) \end{array}$

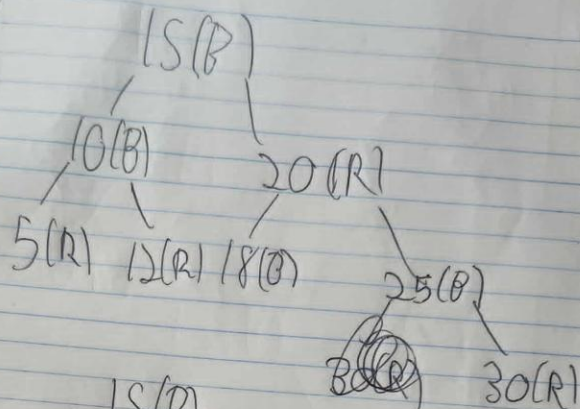
8. $\begin{array}{c} (10, 20, 30) \\ / \quad | \quad \backslash \\ 5 \quad 15 \quad 25 \quad (35, 40) \end{array}$

9. $\begin{array}{c} (10, 20, 30) \\ / \quad | \quad \backslash \\ 5 \quad 10 \quad 15 \quad (35, 40, 45) \end{array}$

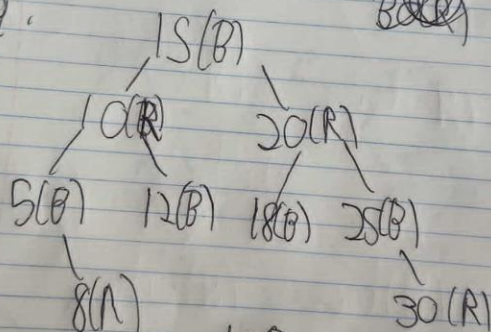
D. Red Black Trees:



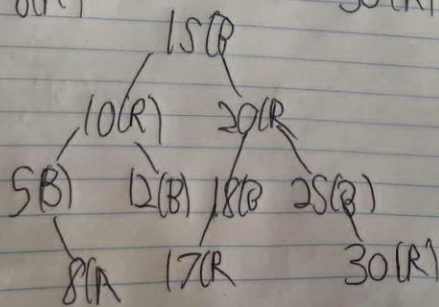
8



9.



10.



Problem 4:

Insertions:

```
▼ TERMINAL
Inserted 2 at 5. Tree height: 0
Inserted 3 at 7. Tree height: 1
Inserted 5 at 15. Tree height: 1
Inserted 7 at 20. Tree height: 2
Inserted 4 at 25. Tree height: 2
Inserted 7 at 17. Tree height: 2
Inserted 10 at 32. Tree height: 3
Inserted 10 at 44. Tree height: 3
Inserted 3 at 48. Tree height: 3
Inserted 5 at 50. Tree height: 3
Inserted 21 at 62. Tree height: 3
Inserted 29 at 78. Tree height: 3
Inserted 6 at 88. Tree height: 3
Inserted 40 at 62. Tree height: 3
Inserted 7 at 90. Tree height: 3
Tree after insertions:
      90:7
    88:6
      78:29
  62:40
    50:5
    48:3
      44:10
  32:10
    25:4
    20:7
      17:7
    15:5
      7:3
      5:2
```

Deletions:

```
Deleted key 32. Tree height: 3
Deleted key 88. Tree height: 3
Deleted key 90. Tree height: 3
Tree after deletions:
      78:29
    62:40
      50:5
      48:3
        44:10
  25:4
    20:7
      17:7
    15:5
      7:3
      5:2
```


Problem 5:n

Algorithm	Random Array	Sorted Array	Reversed Array	Time complexity
Quick Sort	237 ms	262 ms	231 ms	$n \log n$
Merge Sort	22 ms	36 ms	24 ms	$n \log n$
Heap Sort	4 ms	3 ms	3 ms	$n \log n$

```
Merge Sort, random array: 0.021601200103759766
Merge Sort, sorted array: 0.03637290000915527
Merge Sort, reverse array: 0.024237394332885742
Quick Sort, random array: 0.23738741874694824
Quick Sort, sorted array: 0.26210498809814453
Quick Sort, reverse array: 0.23114609718322754
Heap Sort, random array: 0.00395965576171875
Heap Sort, sorted array: 0.0027129650115966797
Heap Sort, reverse array: 0.0027222633361816406
```