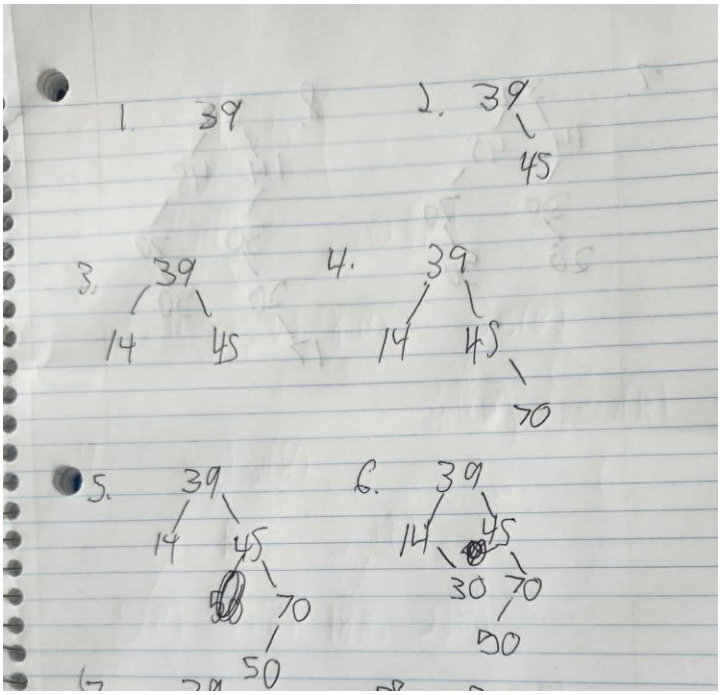
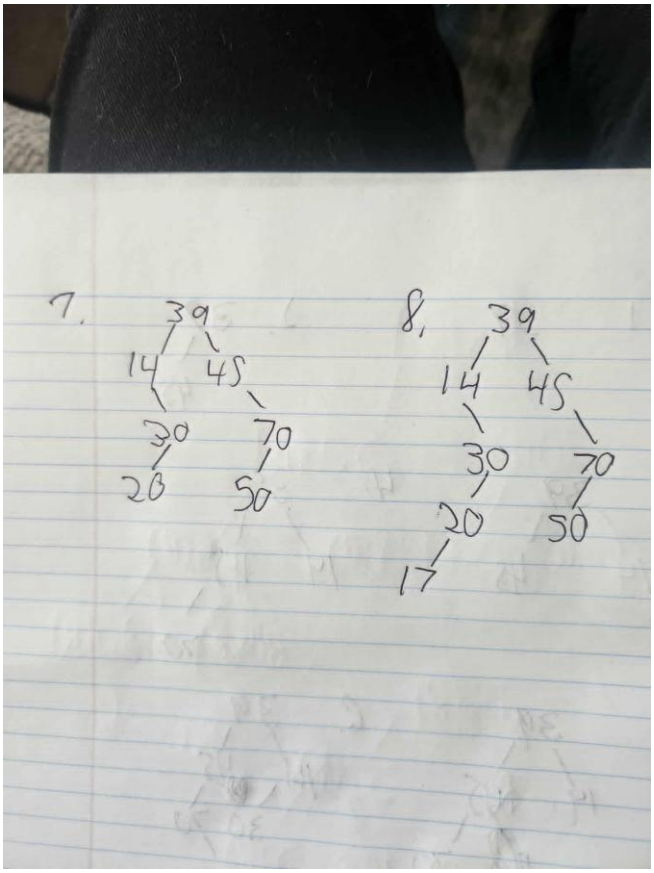
Part 1

Problem 1:



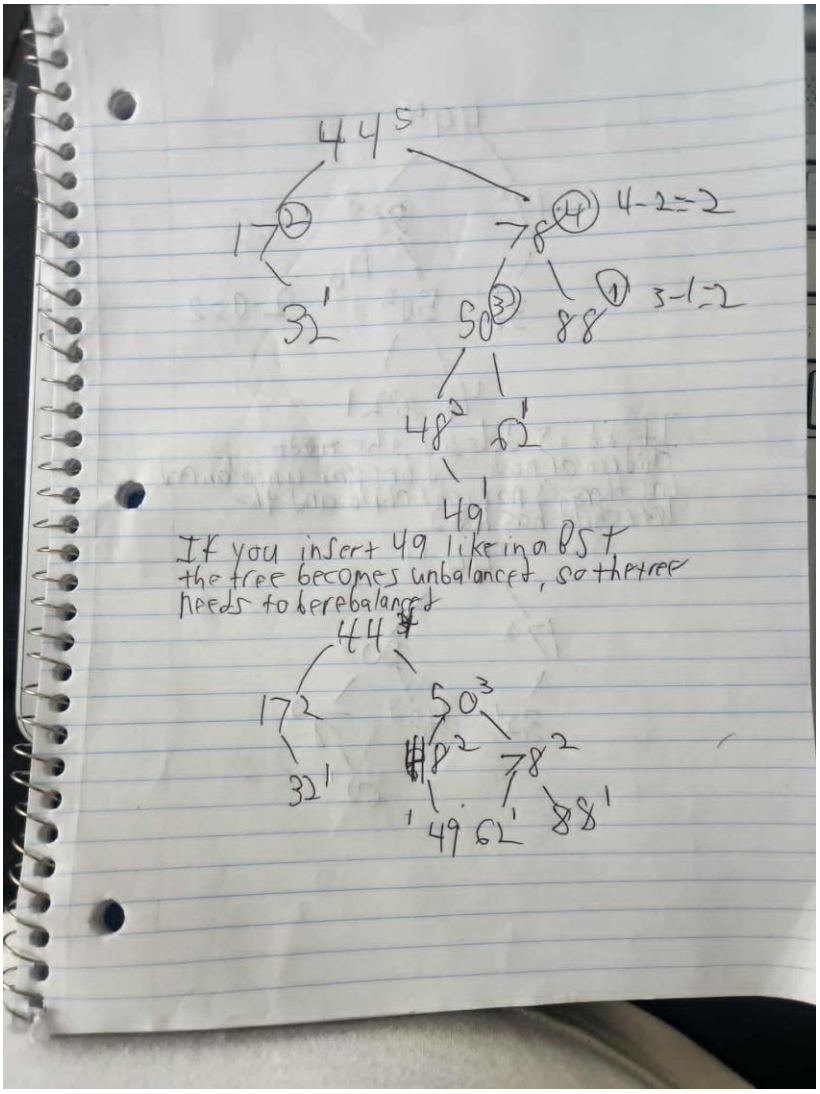


Part 2

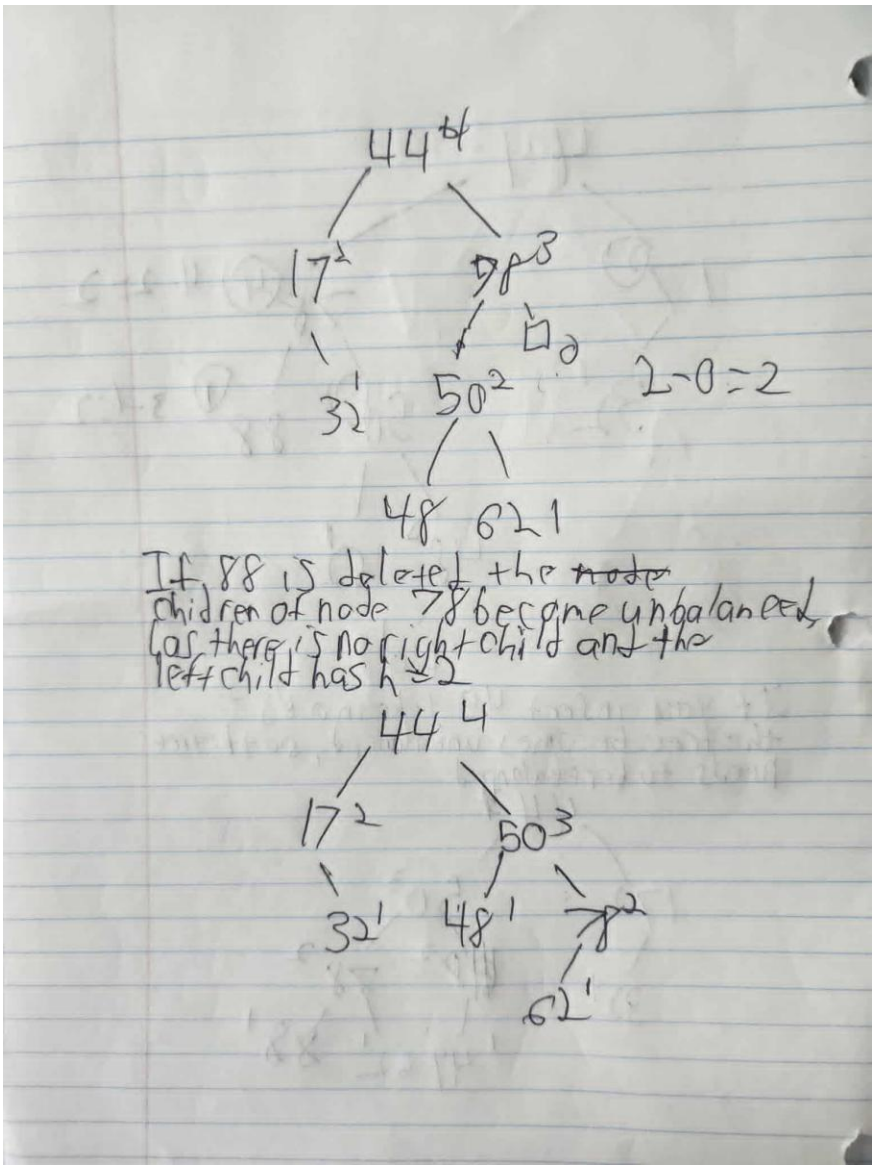
Problem 3:

A. AVL Trees:

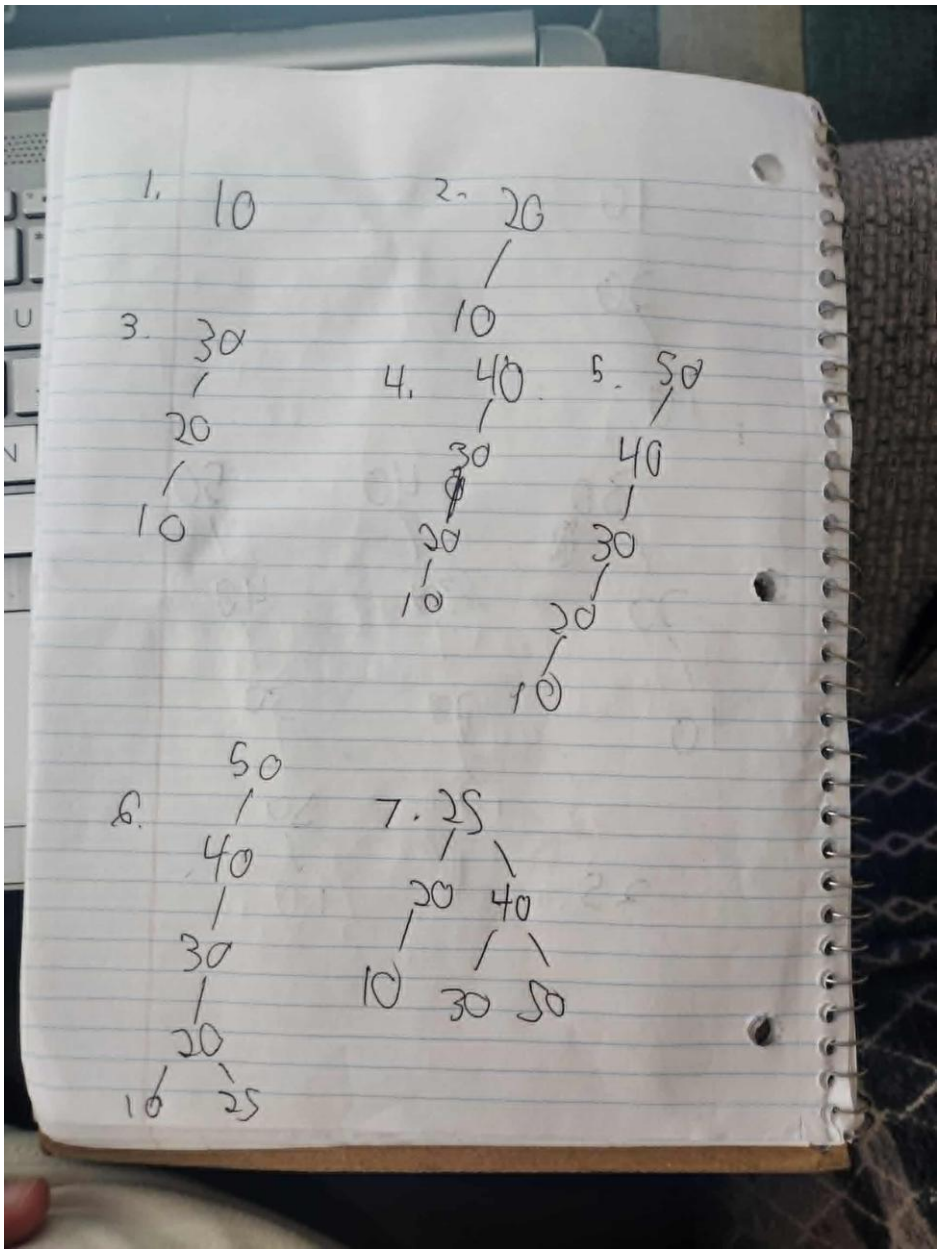
Task 1:



Task 2:

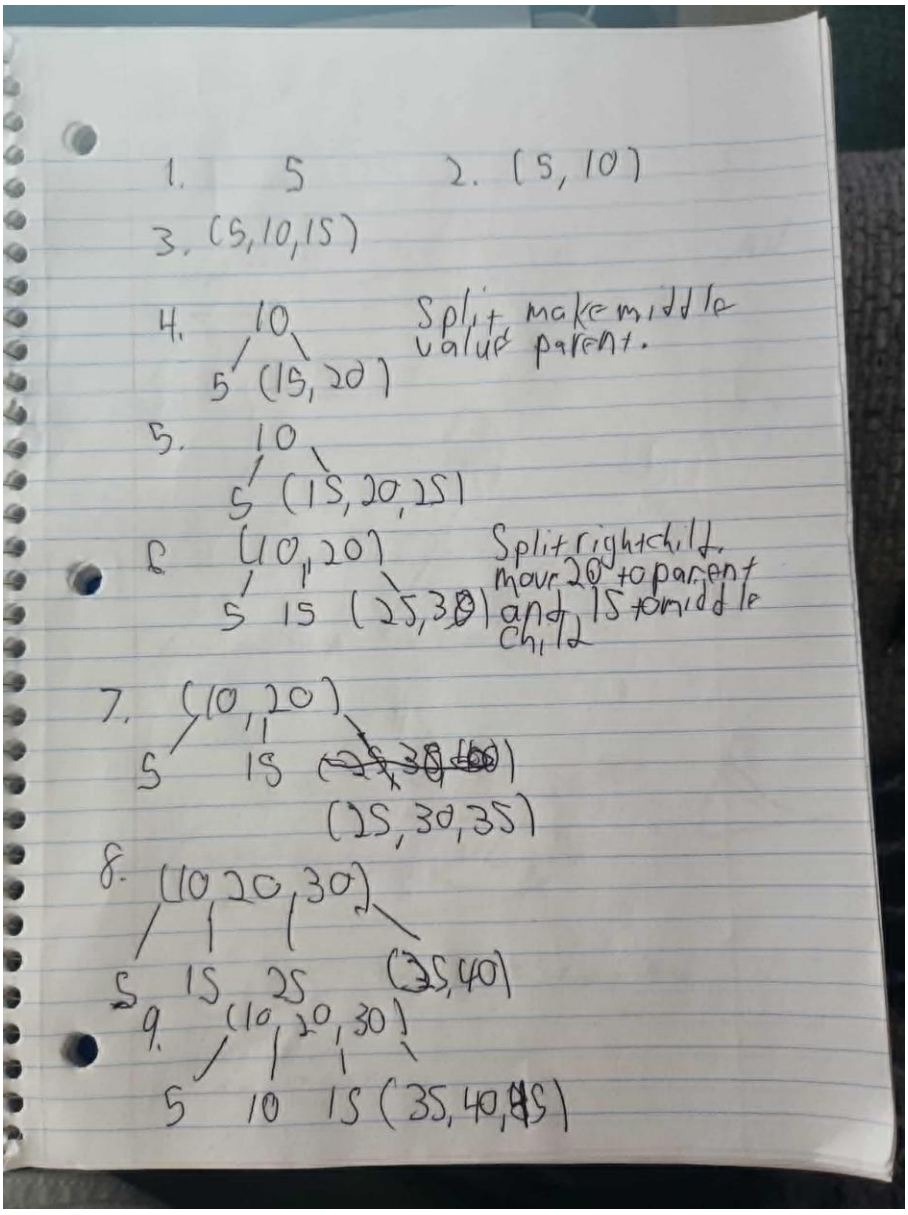

44 4
17 2
78 3
32 1
50 2
2-0=2
48
62 1
If 88 is deleted the children of node 78 became unbalanced, as there is no right child and the left child has h=2
44 4
17 2
50 3
32 1
48 1
78 2
62 1

B. Splay Trees

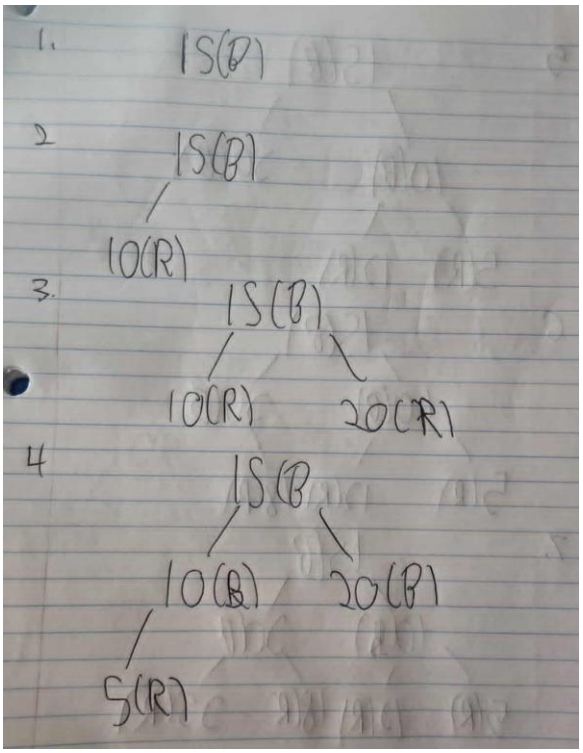


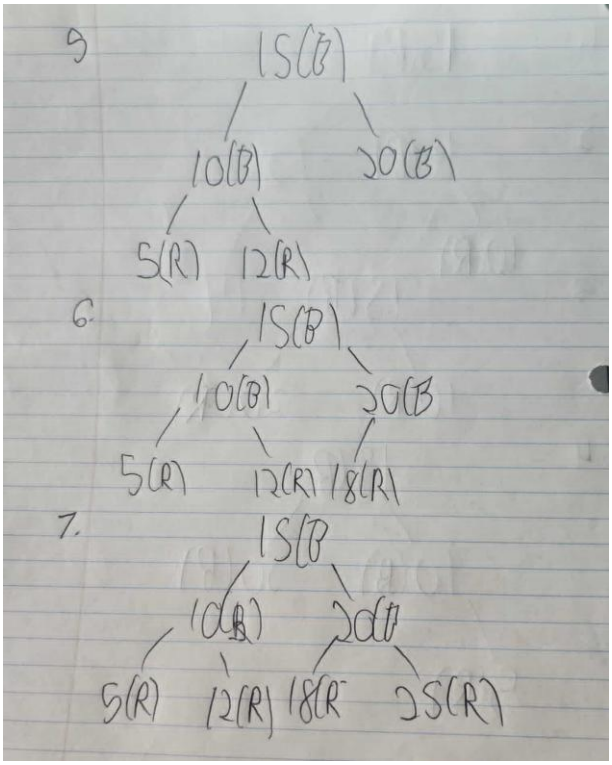
Since 25 was the last key inserted, it is already at the root and the tree doesn't have to be reorganized when it is searched for. If you were to search for another key however, the tree would be reorganized around that key being the root.

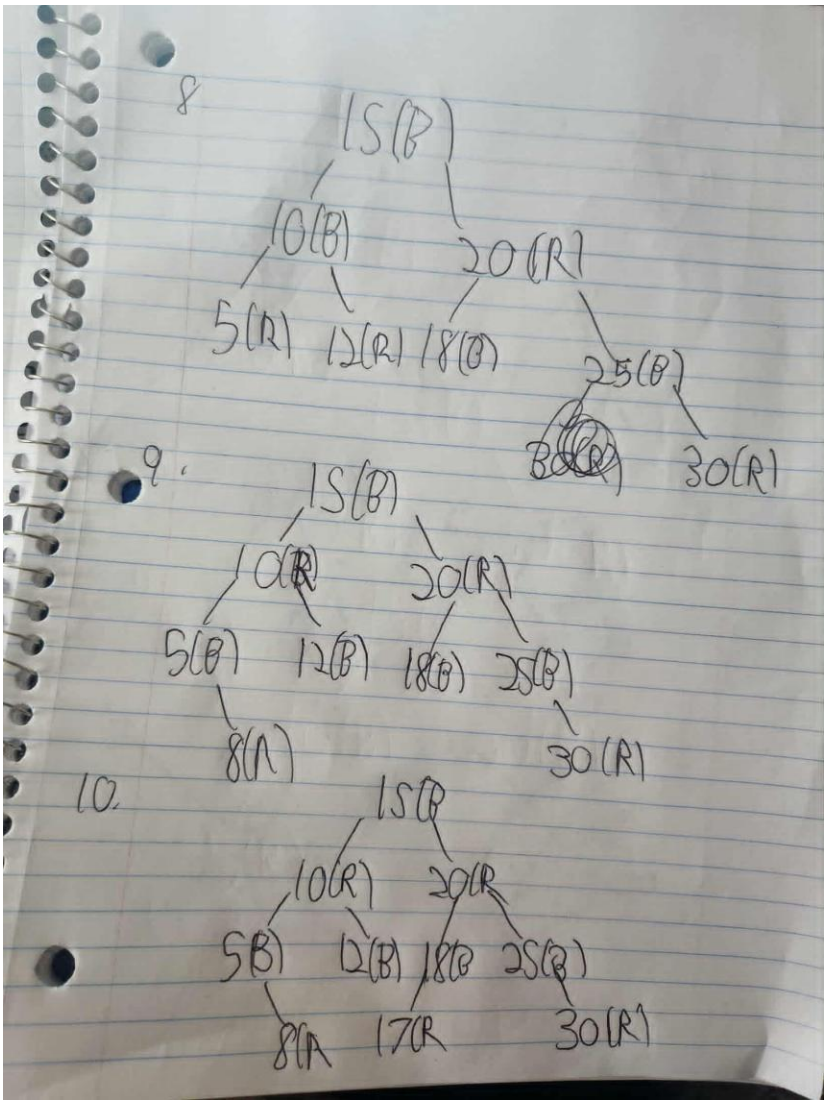
C. 2,4 Trees


1. 5
2. (5, 10)
3. (5, 10, 15)
4. 10
5 (15, 20)
Split make middle value parent.
5. 10
5 (15, 20, 25)
6. (10, 20)
5 15 (25, 30)
Split right child. move 20 to parent and 15 to middle ch, 12
7. (10, 20)
5 15 (25, 30, 35)
8. (10 20, 30)
5 15 25 (35, 40)
9. (10, 20, 30)
5 10 15 (35, 40, 45)

D. Red Black Trees:

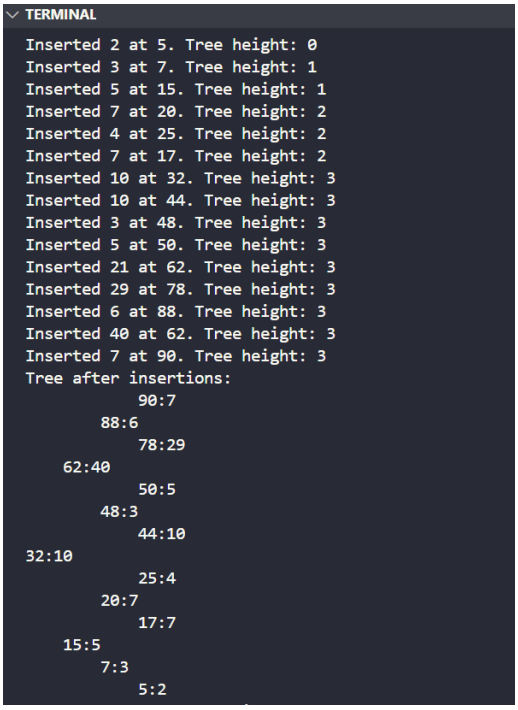




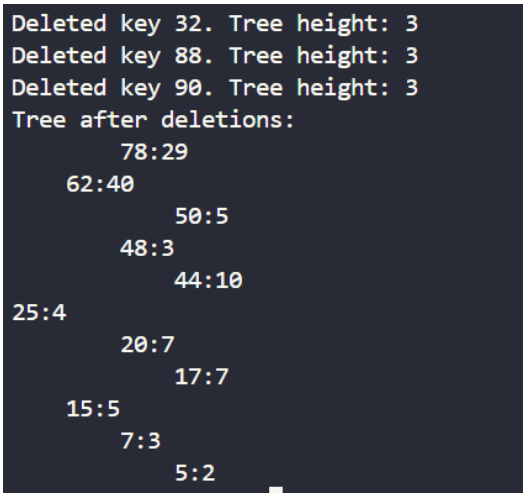

8
15(B)
10(B)
20(R)
5(R)
12(R)
18(B)
25(B)
30(R)
9.
15(B)
10(R)
20(R)
5(B)
12(B)
18(B)
25(B)
8(R)
30(R)
10.
15(B)
10(R)
20(R)
5(B)
12(B)
18(B)
25(B)
8(R)
17(R)
30(R)

Problem 4:

Insertions:

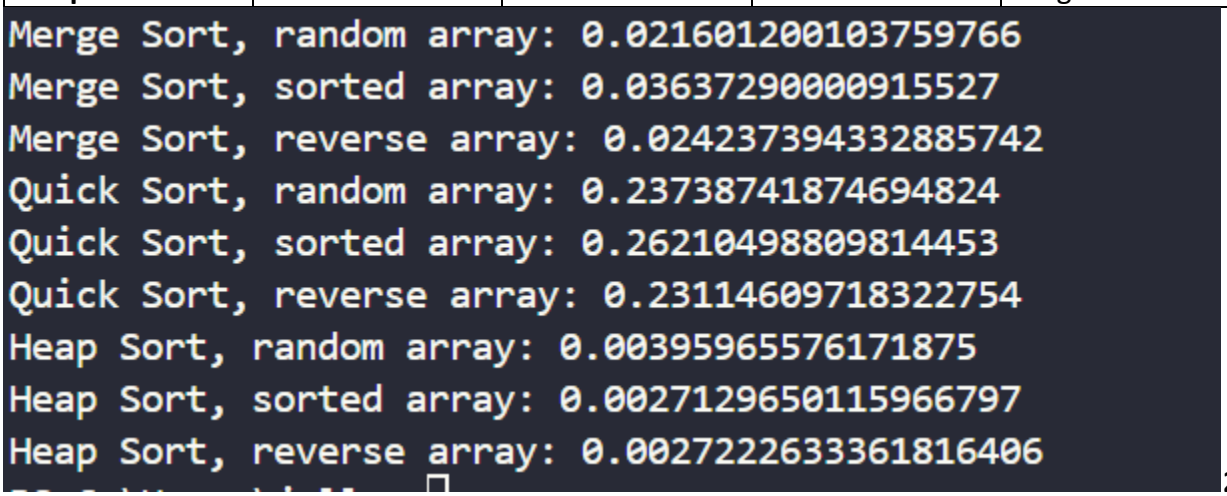


Deletions:



Problem 5:n

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



2