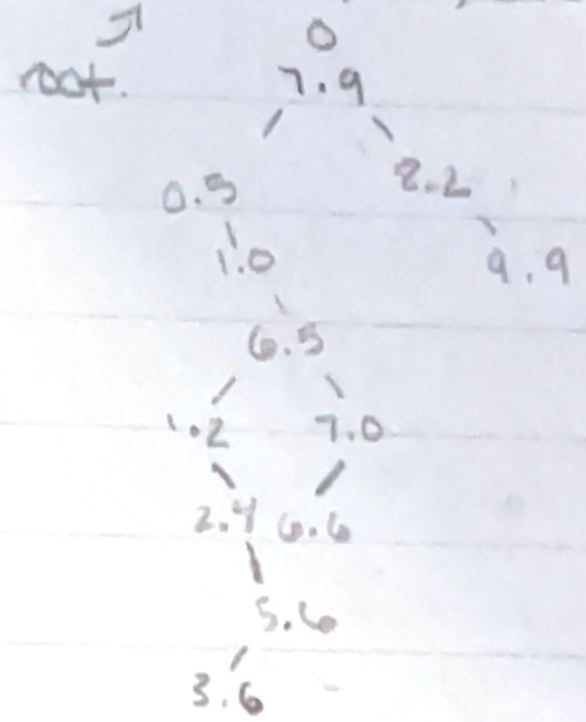
Elyas Ata

Trees & heaps

1.)

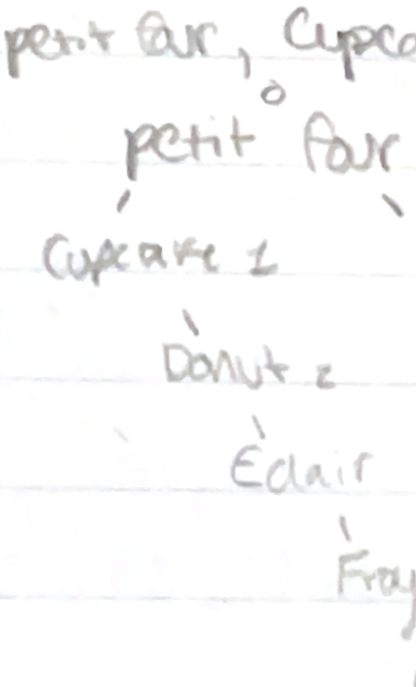
a.) [7.9, 0.5, 1.0, 6.5, 8.2, 7.0, 6.6, 9.9, 1.2, 2.4, 5.6, 3.6]

root.



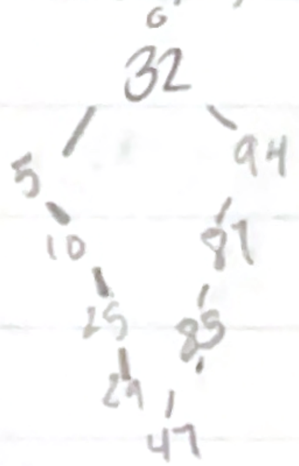
height: 7 because longest from root down left side sub tree

b.) [petit four, Cupcake, Donut, Eclair, Frayo, Gingerbread, Honeycomb]



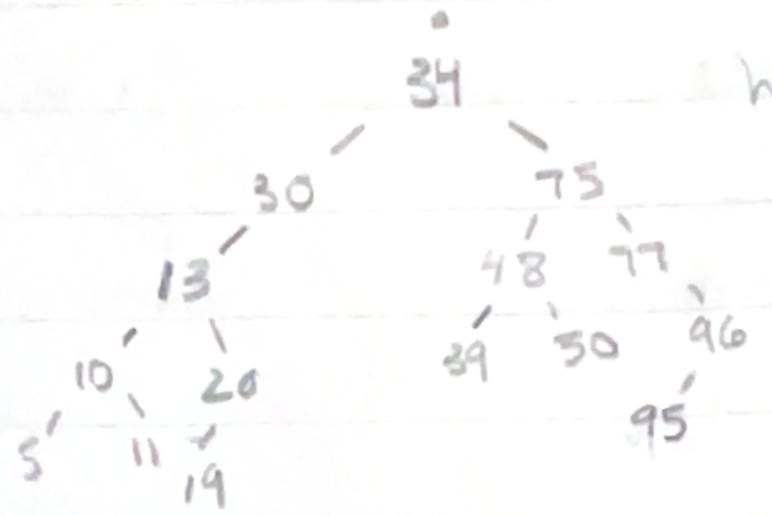
height: 6 because left subtree.

c.) [32, 5, 94, 87, 10, 85, 47, 25, 24]



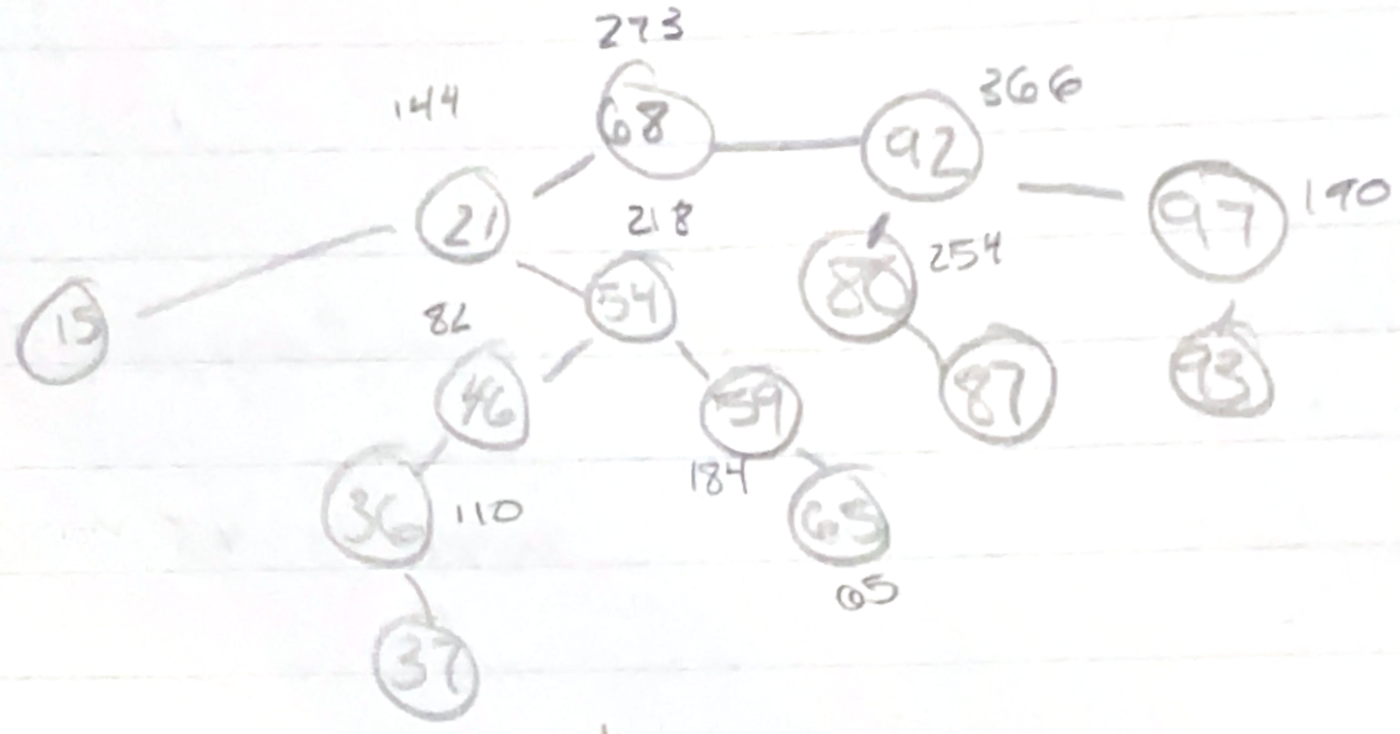
height: 4

d.) [34, 30, 75, 77, 96, 48, 39, 50, 93, 13, 10, 5, 11, 20, 19]



height = 4

2.)



it goes root left right

preorder: 68, 21, 15, 54, 46, 36, 37, 59, 65, 92, 80, 87, 97, 93

68 = 21, 92 = 68 + 21 + (92 * 2) = 273 / 270

21 = 21 + 15 + (54 * 2)

15 = 15 + 0 + (0 * 2) = 15

54 = 54 + 46 + (59 * 2) = 218 / 220

46 = 46 + 36 + (0 * 2) = 82

36 = 36 + 0 (37 * 2) = 110

37 = 37

59 = 59 + 0 (65 * 2) = 189 / 190

65 = 65

92 = 80 (97 * 2) = 366

80 = 0 + (87 * 2) = 254

87 = 87 + 0 + 0 = 87

97 = 93 + (0 * 2) = 190

93 = 93

Elyas Atta Trees and heaps

2) inorder = left root right

15 = 0 (0x2) = 15

21 = 21 + 15 (54 x 2) = 144

36 = 0 (37 x 2) = 110

37 = 37

46 = 46 + 110 (0 * 2) = 46 + 110 = 156

54 = + 156 + (59 x 2) = 328

59 = + 0 + (65 * 2) = 189

65 = 65

68 = + 144 + (92 * 2) = 396

80 = + 0 + (87 x 2) = 254

87 = + 87

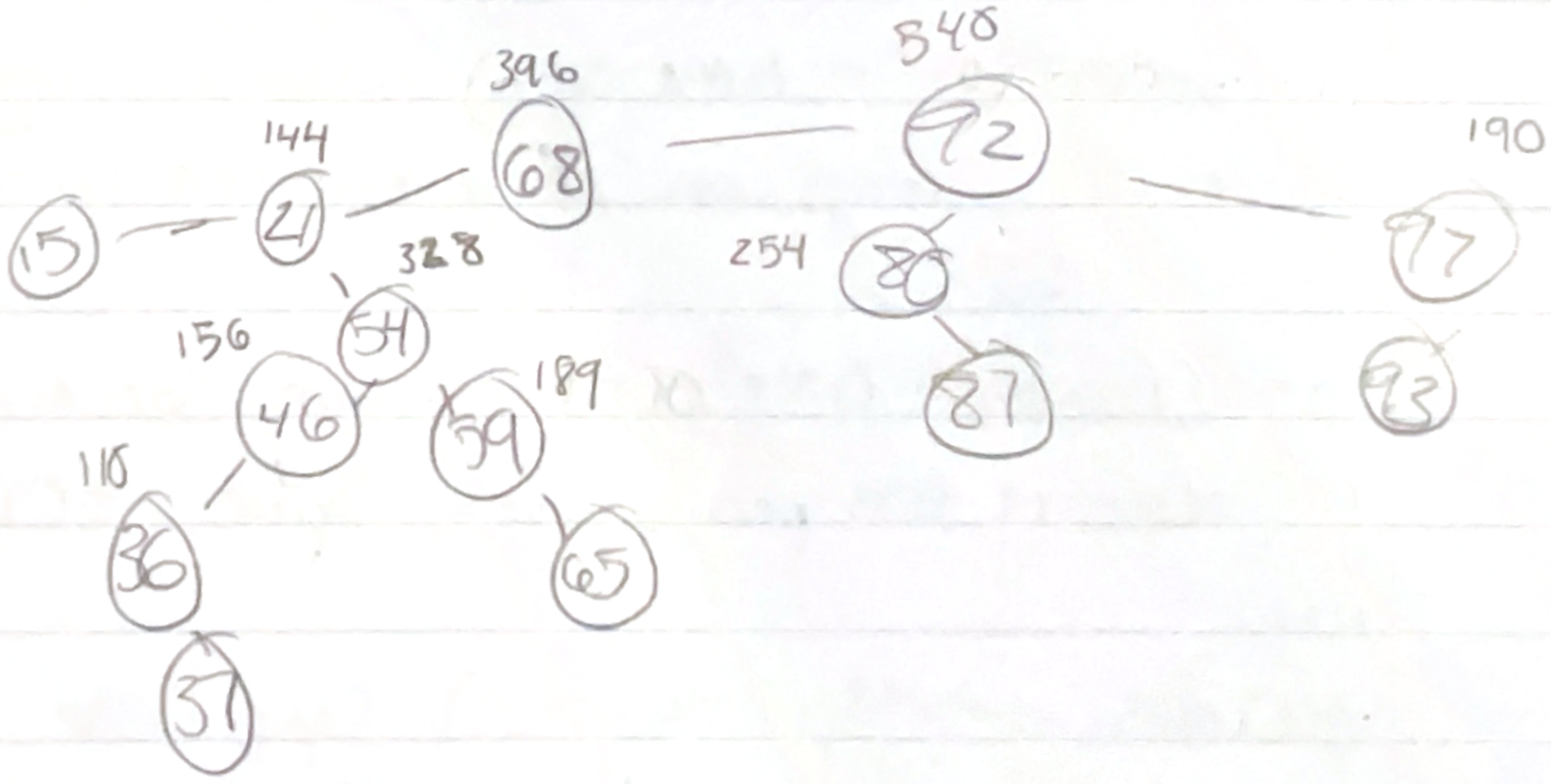
92 = + 254 + (97 x 2) = 540

93 = 93

97 = + 93 + 0 (0 x 2) = 190

2 b) no, neither of the tree from preorder and inorder because of rule left child < node < right child

2c) none are AVL because it needs to be BST first balance factor is not met violates -1, 0, 1 because of 3.



5.) initaize(candidates (LinkedList<String> candidates)

- O(N Log N) add each name to hashmap.

inserting into a priority heap takes nlogn

Space: we store N votes and heap, names stores n because the number of candidates grows.

- setElectoralSize (int p)

time O(1) stors single number constant time

Spac O(1) only stors 1 at a time fixed amount

-CastVote(String candidate)

time O(n) finding the candidate in hashmap and removing candidate

Space O(1) only changing vote count and rearranging the map

- castRandomVote - time O(n) randomly picking from list

Spac O(1) relies on castVote picks random

- ng Election - time O(n) removing specific candidate and removing from priority queue. Space - O(1) keeps changing map.

-getTopKCandidates time O(n) Cop, heap and OnLogn to remove top element k times.

audit Election

time o(nLogn)

copying heap is o(n) and remove is O(n)

Spae o(n) to hold all elements