CSE 101-02 2-28-23

- Midterm 2 Thursday

Runtime of tree operations:

Traversals

InOrderTreeWalk(x)

PreOrderTreeWalk(x)

PostOrderTreeWalk(x)

cost = $\Theta(n)$, where n = # nodes.

Queries

root

TreeSearch(x)

" Min(x)

" Max(x)

" Successor(x)

" Predecessor(x)

Depth

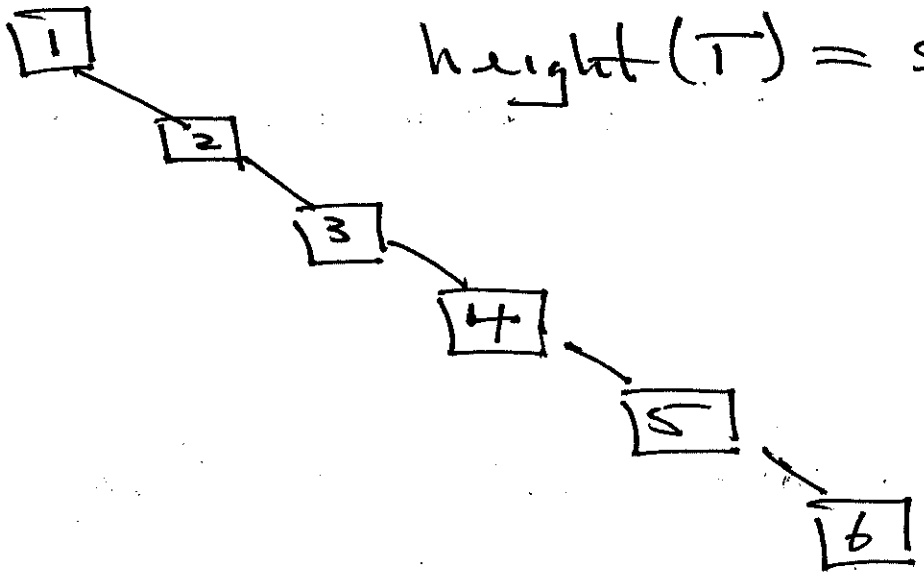
height(T) = dist. from root to the deepest leaf

height(x) = height(sub-tree rooted at x)

note: height(x) ≤ height(T)

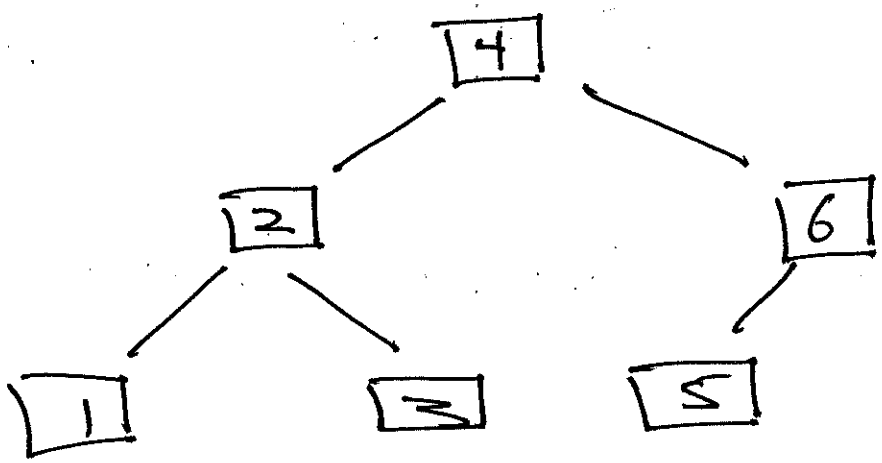
worst case cost of queries is $\Theta(\text{height}(T))$

Ex worst possible BST



height(T) = 5

Ex. Best possible BST



height(T) = 2

In general

$$\log n \leq \text{height}(T) \leq n-1$$

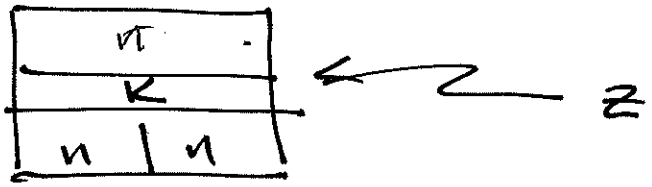
we call T <u>balanced</u> iff

height $\approx \log n$.

12.3 Insertion & Deletion

To insert new node z, and maintain BST properties:

- set key[z] = k
- set z.parent = z.left = z.right = nil



strategy:

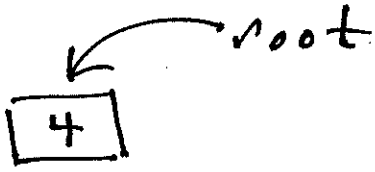
Search for key k in T, find nil child where key k would reside, if it were in T.

Ex. see Pseudo-code for TreeInsert(T, k)

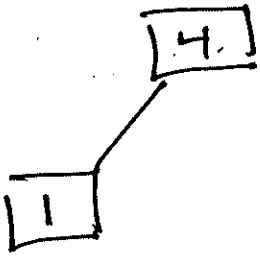
Keys: 4 1 7 2 8 3 5 6

empty state: T.root = nil

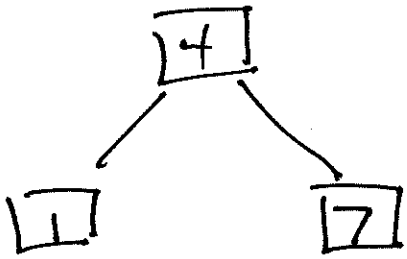
insert (4)



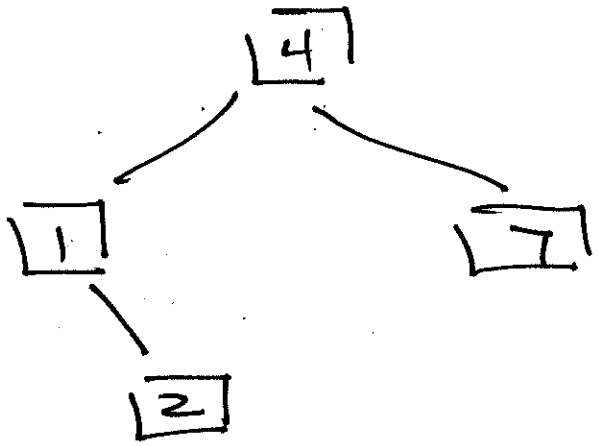
insert (1)



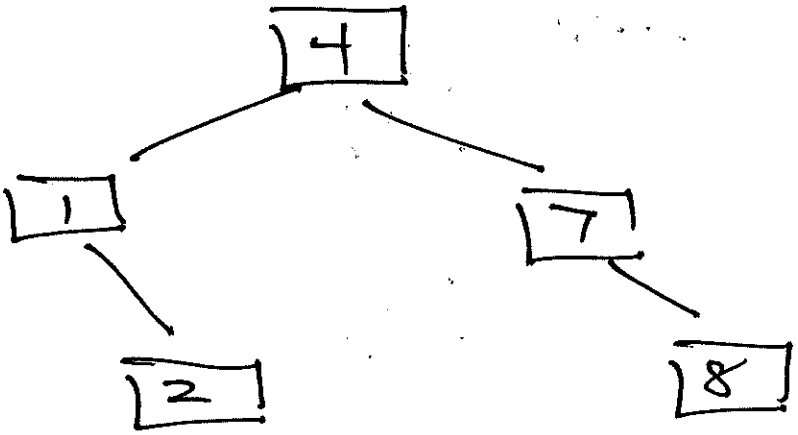
insert(7)



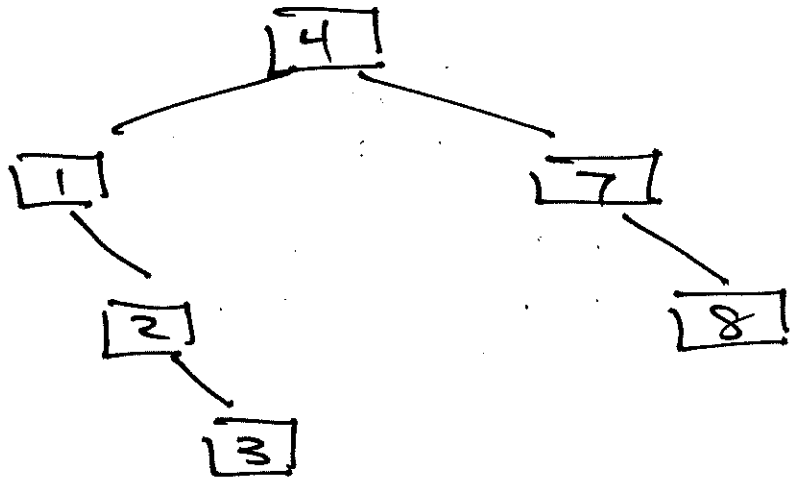
insert(2)



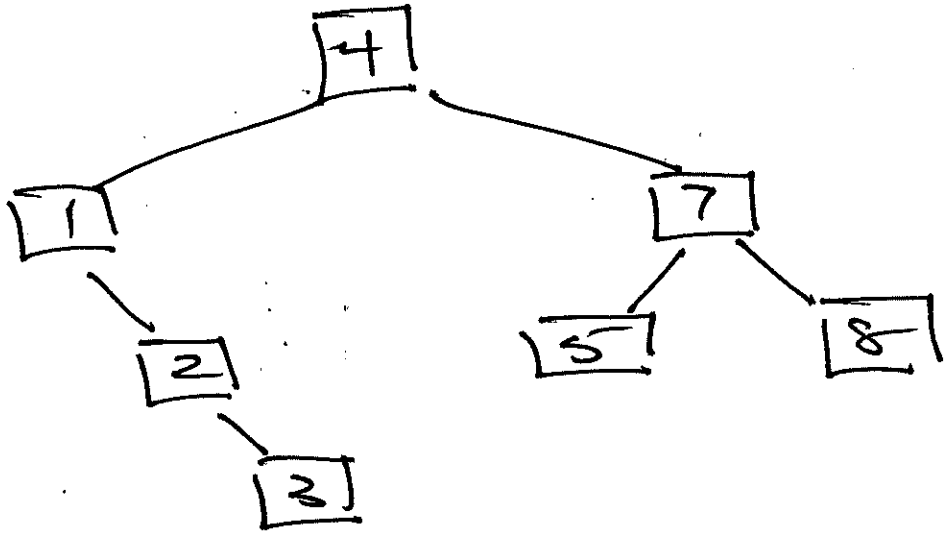
insert(8)



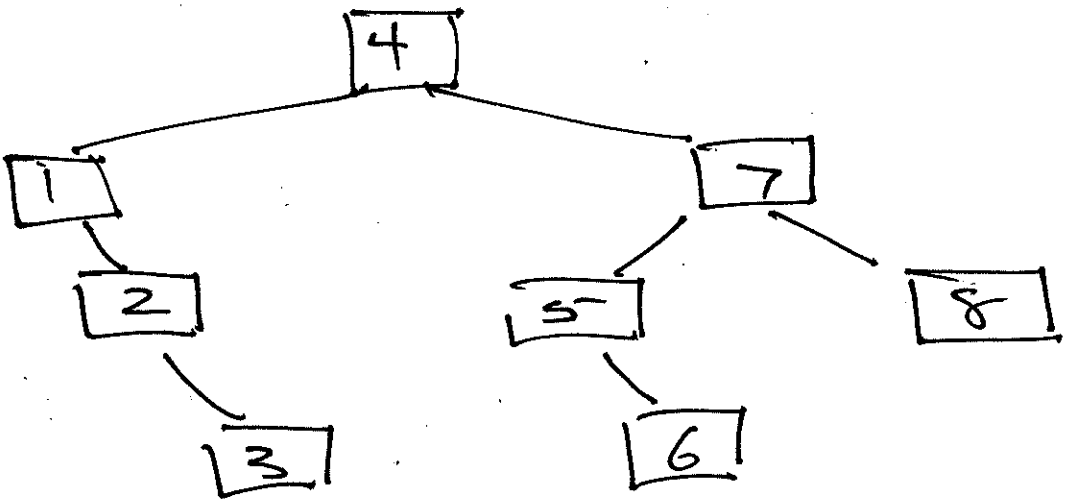
insert(3)



insert(5)



insert(6)



worst case runtime

$\Theta(\text{height}(T))$

Deletion: delete z from T

3 cases:

case 1 : z has no children

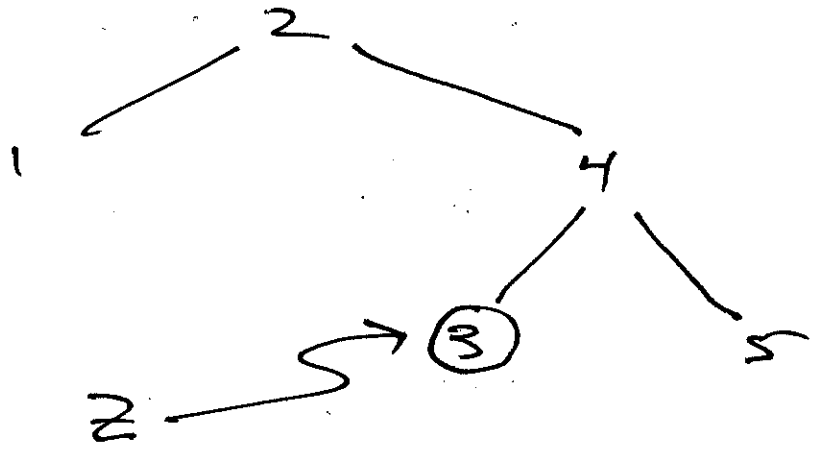
case 2 : z has 1 child

2.1 right but no left

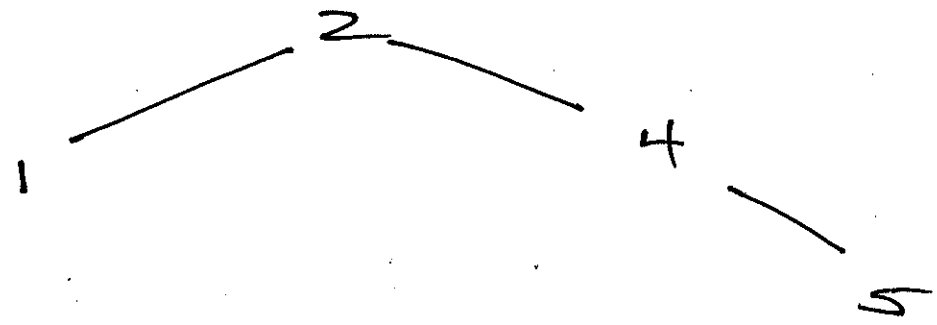
2.2 left but no right

case 3 : z has 2 children

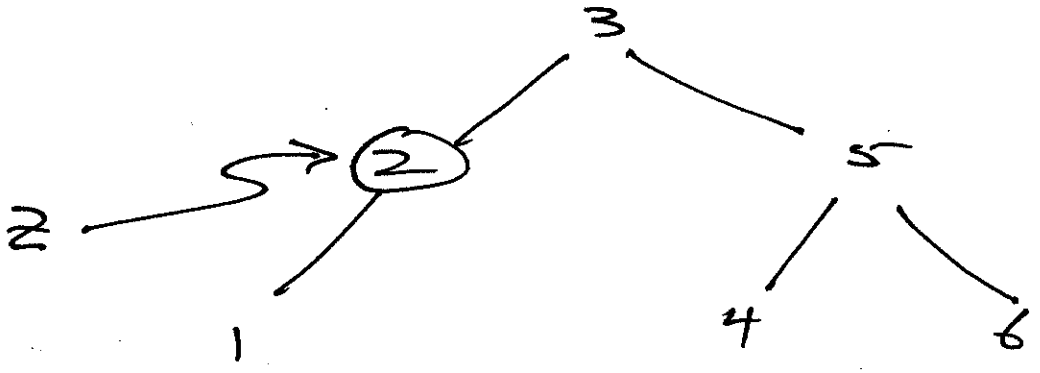
Ex. Case 1



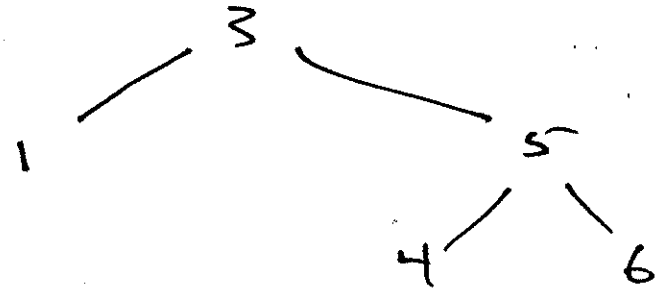
delete(3)



Ex. case 2 (actually 2.2)



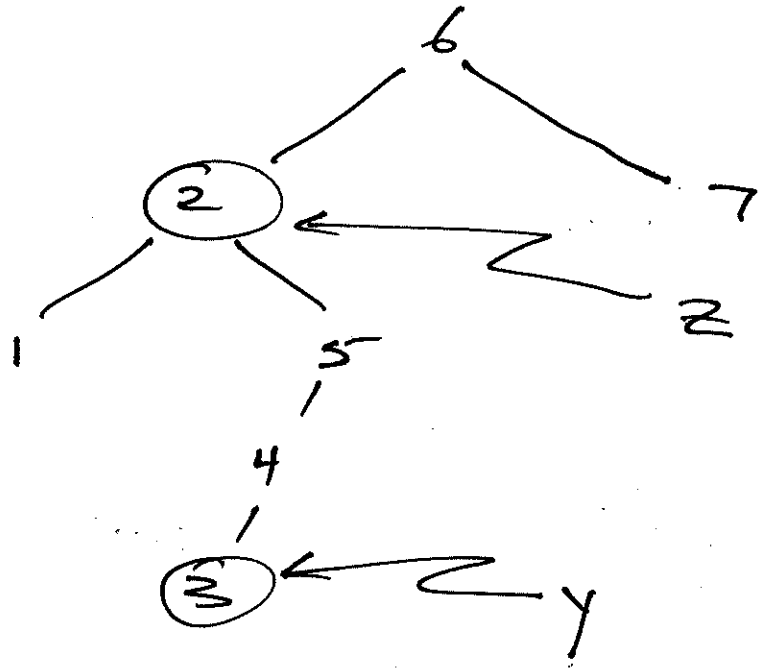
delete(2)



splice 2 out of 'local' linked list.

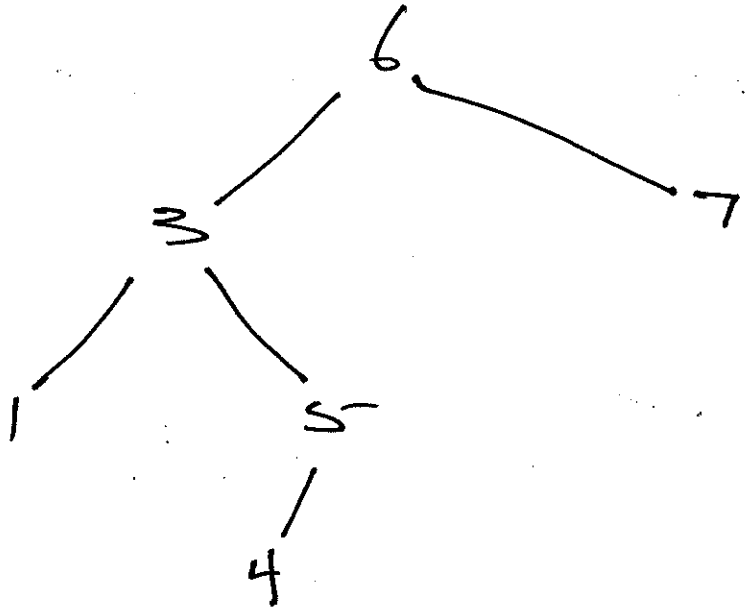
Exercise: draw example of 2.1

Ex. Case 3: z has 2 children

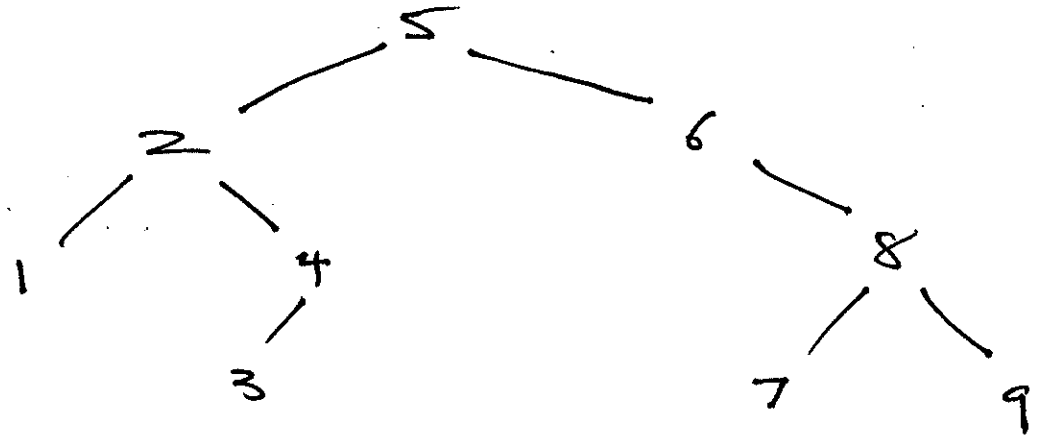


splice out successor y, replace z with y

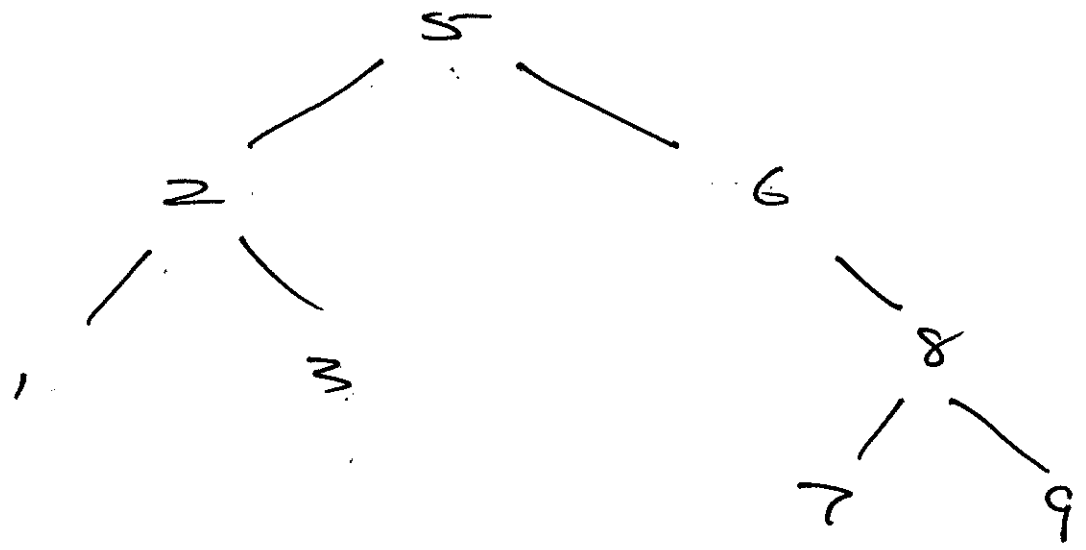
delete(2)

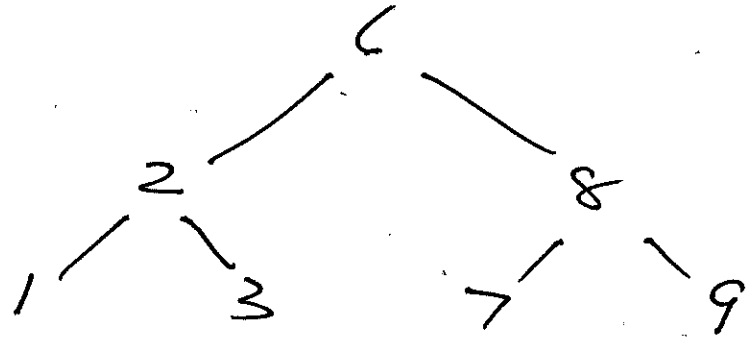


Ex insert : 5 2 6 4 3 8 7 1 9



Delete: ~~4~~ ~~5~~

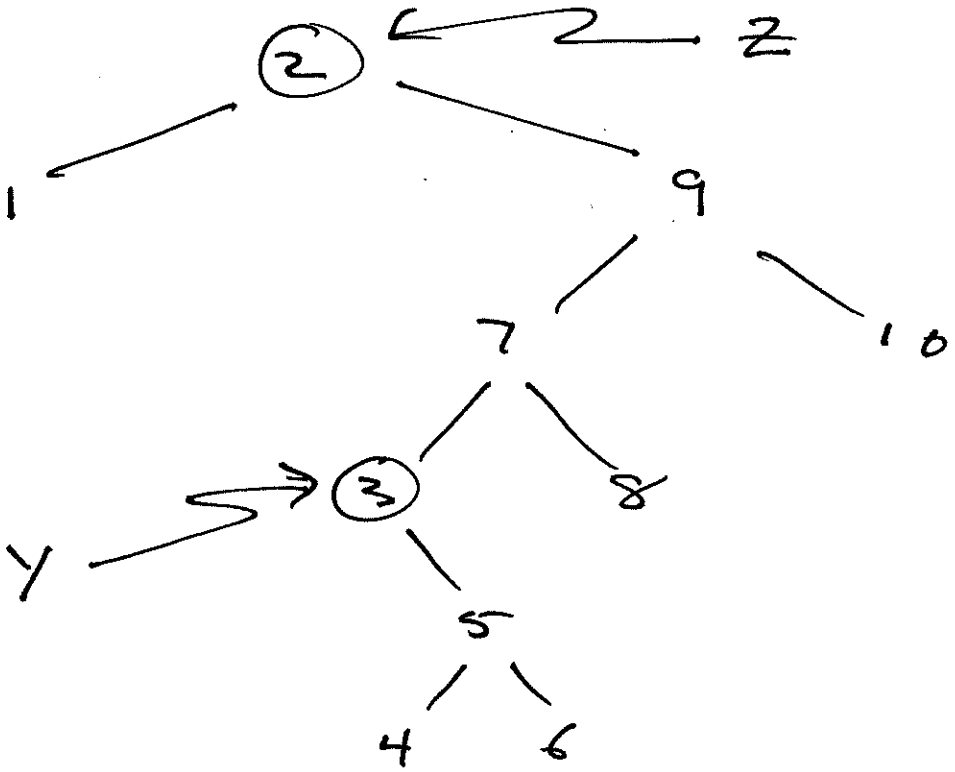


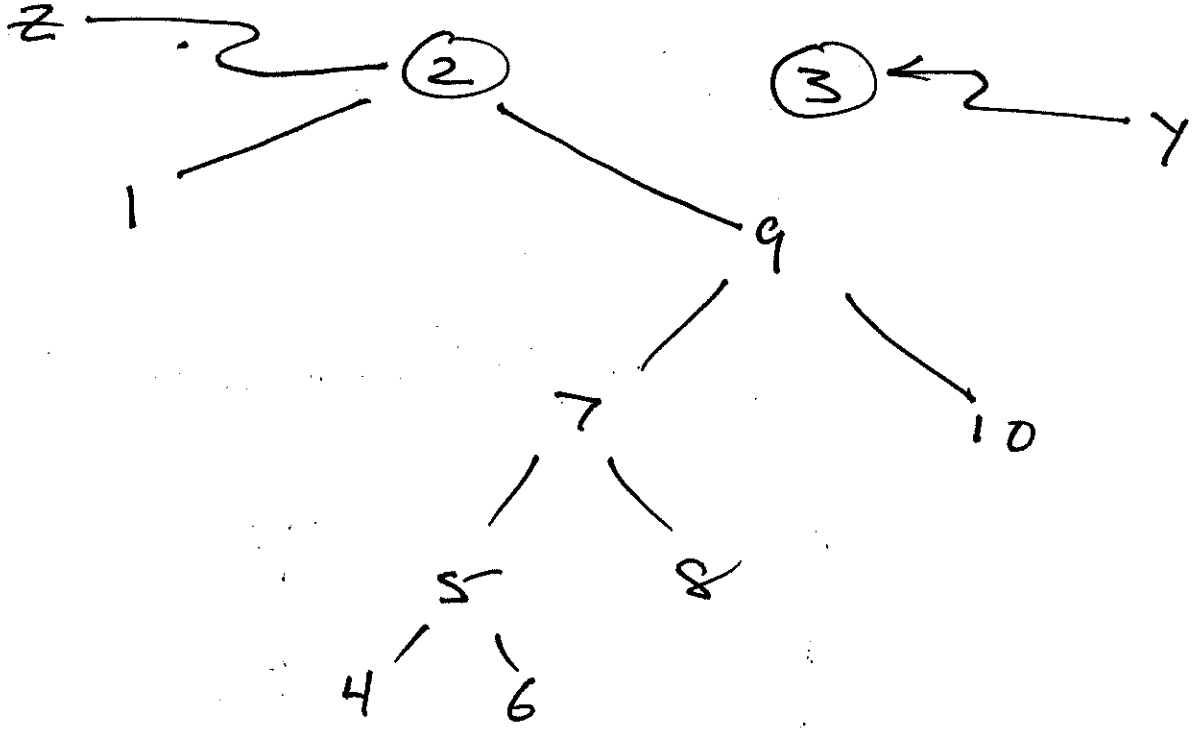


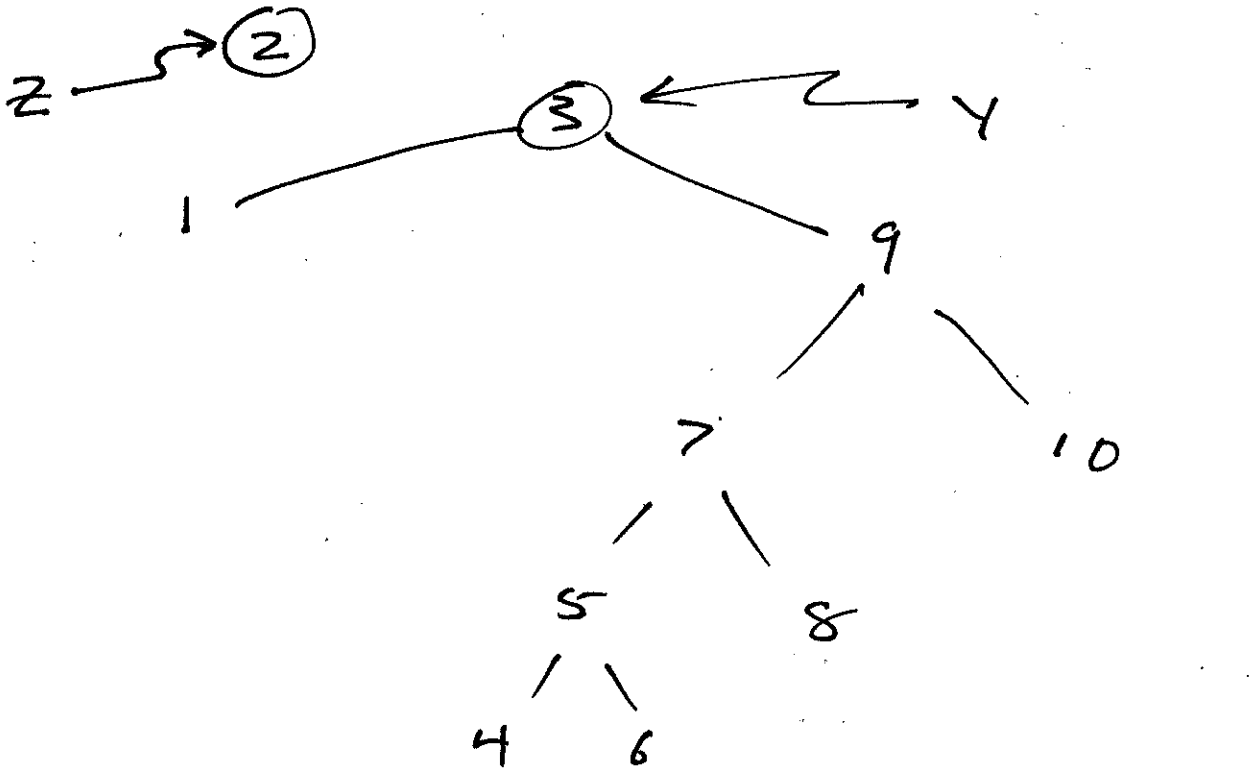
Helper for delete( )

Transplant(T, u, v)

Ex. Case 3




Z
2
3
Y
1
9
7
10
5
8
4
6


Z
2
3
Y
1
9
7
10
5
8
4
6