Pa6: ext. to Friday (last one)

Mid 2: Thur May 25

## Queries

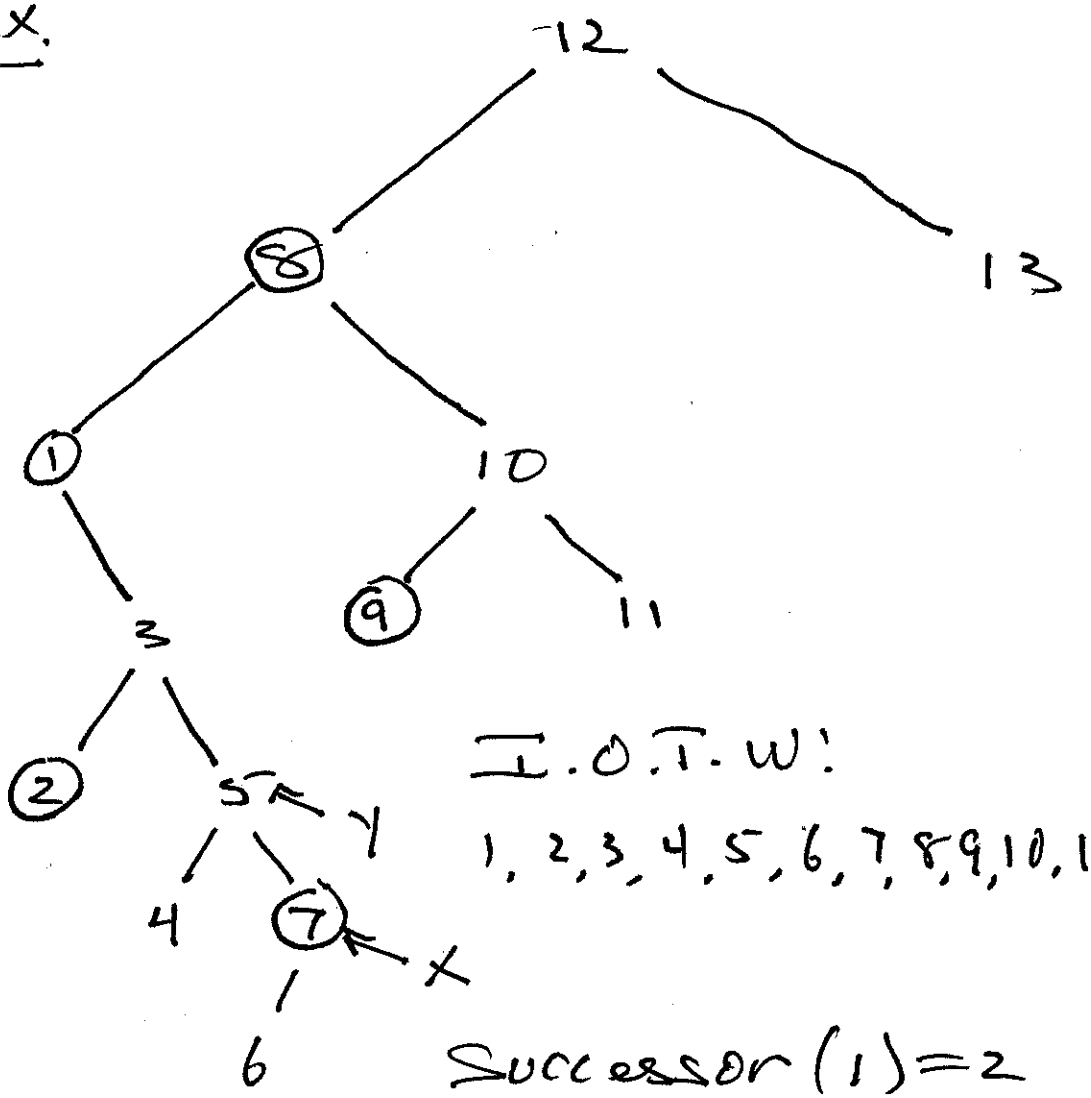
node ↓ key ↓

- TreeSearch(x, k)
- TreeMinimum(x)
- TreeMaximum(x)

## Defn

The successor of a node x is the next node after x to be printed in an I.O.T.W.

Ex.




I.O.T.W:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

Successor(1) = 2

Successor(8) = 9

Successor(7) = 8

Successor(13) = nil

2 cases for successor:

case 1: x has a right child.

The successor of x is the leftmost descendant of that child.

case 2: x has no right child.

The successor of x is the lowest ancestor of x, whose left child is also an ancestor of x.

Exercise:

- define Predecessor of x.
- characterize Predecessor in 2 cases.
- Write Pseudo-code

## Runtimes:

- Traversals: In, Pre, Post - $\Theta$.T.W.
  cost = # recursive calls $= n$
  $= \Theta(n)$.

- Queries: Tree-search, min, max, successor, predecessor

  cost = length of a longest line of ancestory

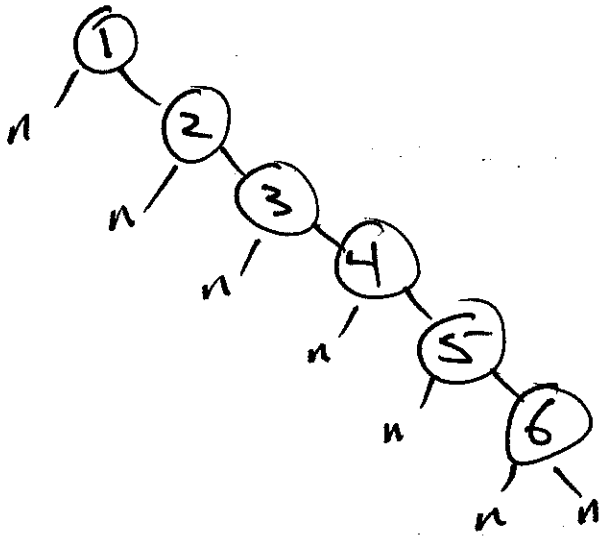
## Defn

- height(T) = distance from root to deepest leaf
- height(x) = height of the subtree rooted at x.

so cost of query (in worst case)

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

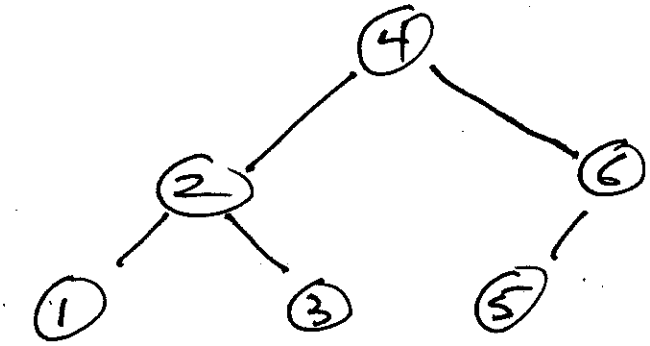
Ex. Worst BST for queries: $n = 6$




cost $= n = \Theta(n)$

height$(T) = 5$

Ex.




height$(T) = 2$

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

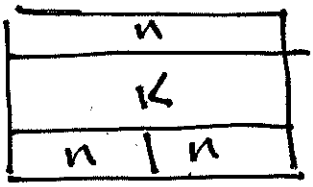
In general: $\log(n) \leq \text{height}(T) \leq n-1$

$\log(n)$: balanced trees

# 12.3 in CLRS: Insertion & Deletion

To insert new node z, and maintain the BST properties:

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




- find a node to adopt z as left or right child.

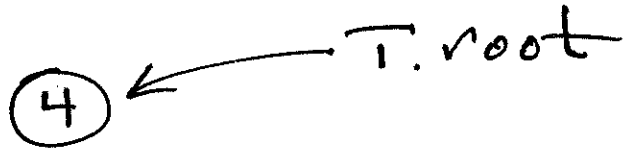
how?

Do a search for key k in T, find nil child where k would live, put z there.

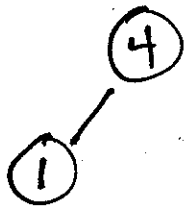
Ex. insert keys: 4, 1, 7, 2, 8, 3, 5, 6

empty tree: T.root = nil

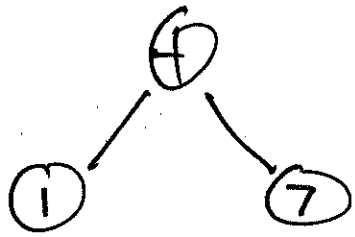
insert(4):



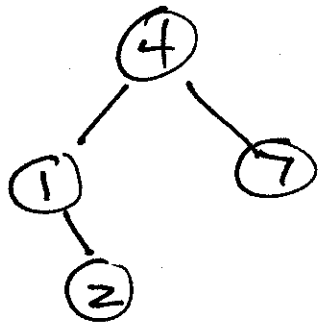

insert(1):



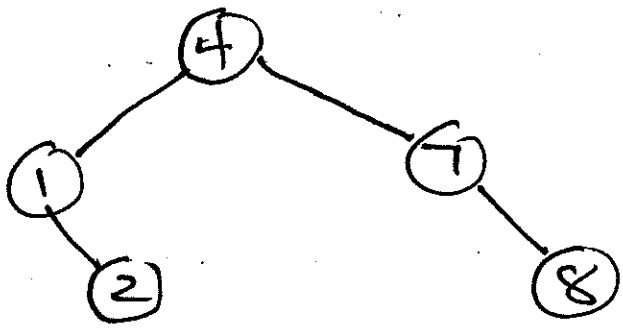

insert(7):



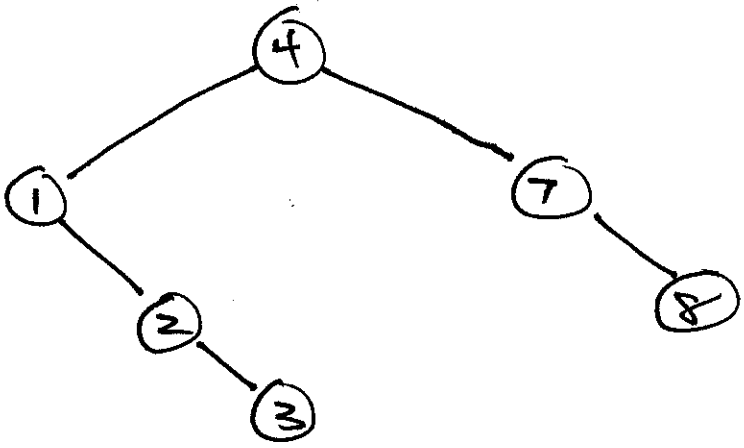

insert(2):



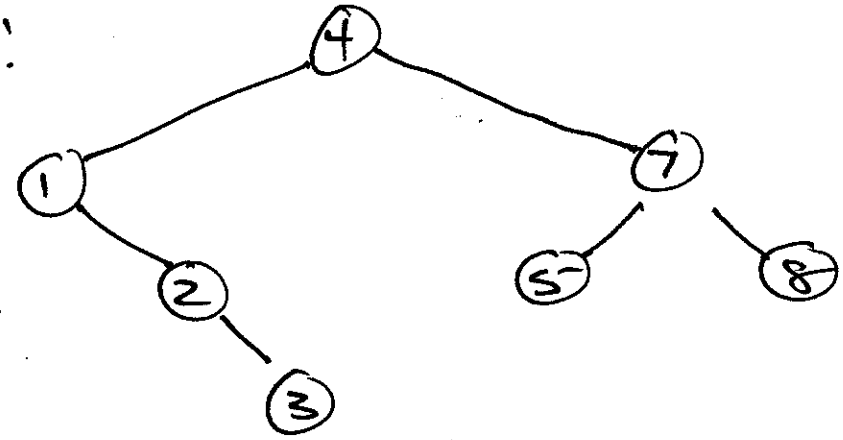
insert(8):



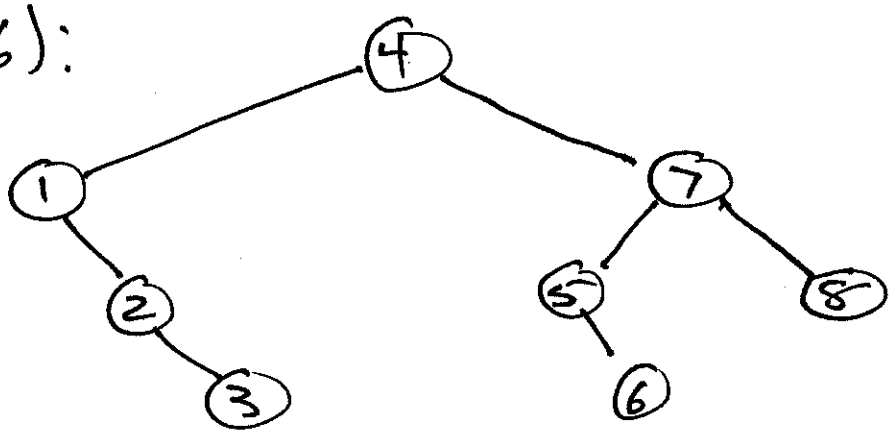

insert(3):




insert(5):




insert(6):



Note:

root left right

Pre. O.T.W : (4), [1, 2, 3], [7, 5, 6, 8]

Post. O.T.W : [3, 2, 1], [6, 5, 8, 7], (4)

left right root

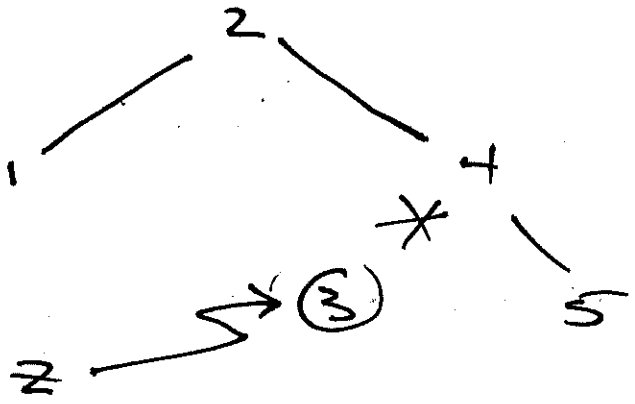
- insert Keys in Pre-order reproduces BST.
- insert Keys in reverse Post-order reproduces BST.

# Deletion: 3 cases

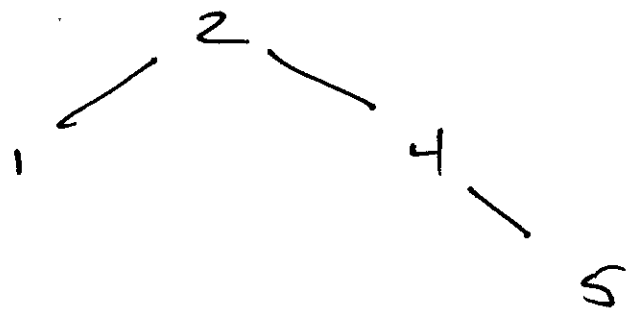
goal: delete z from T

## case 1: z has no children

Ex.



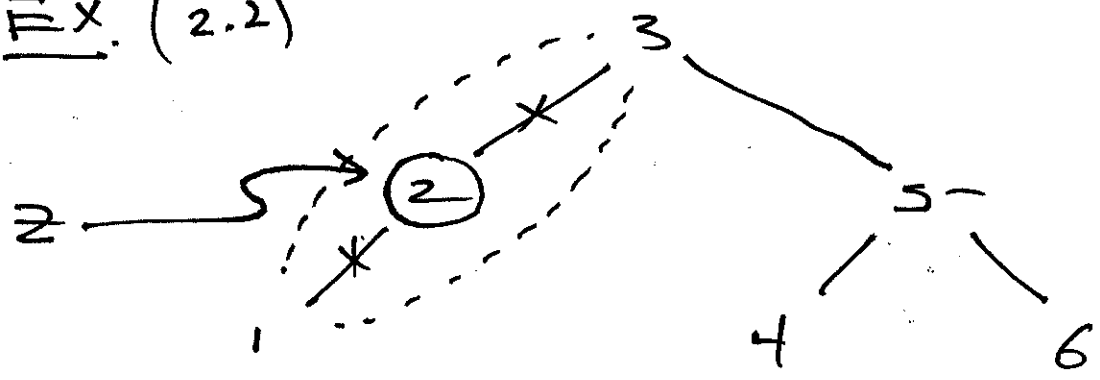

result:



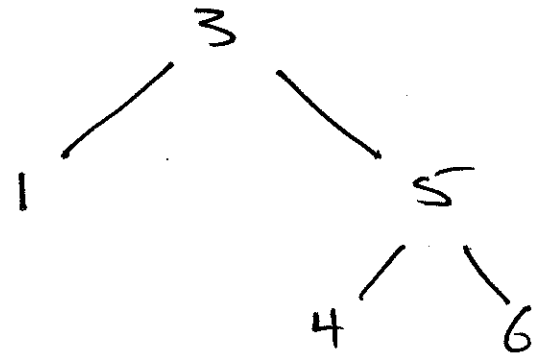
Case 2: z has 1 child.

Subcases: 2.1: z has right but no left
2.2: " " Left " " right

Ex. (2.2)



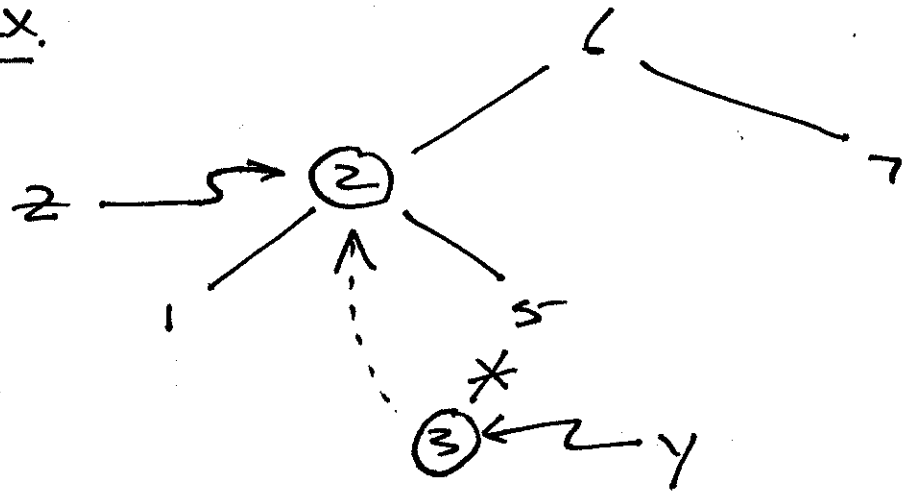

Result:



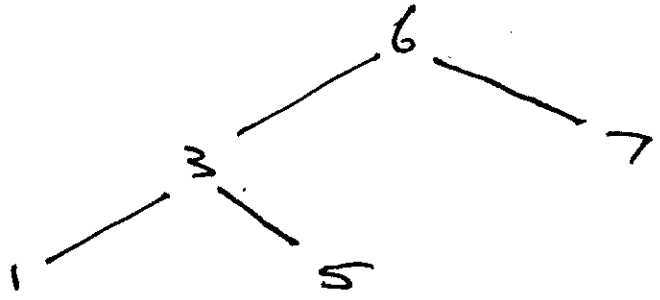
case 3: z has 2 children.

splice out z's successor y (which has no left child), then replace z by y.

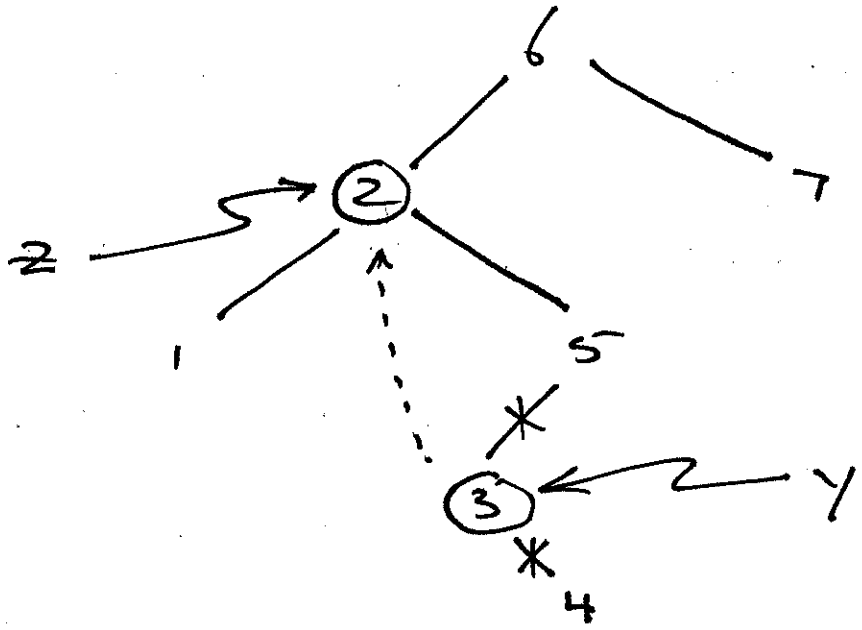
Ex.



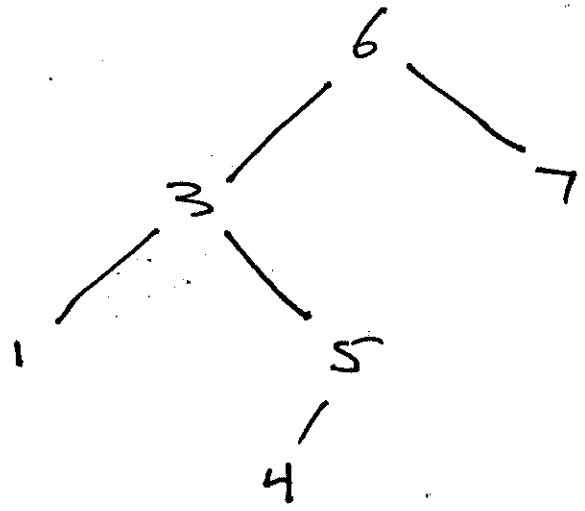

result:



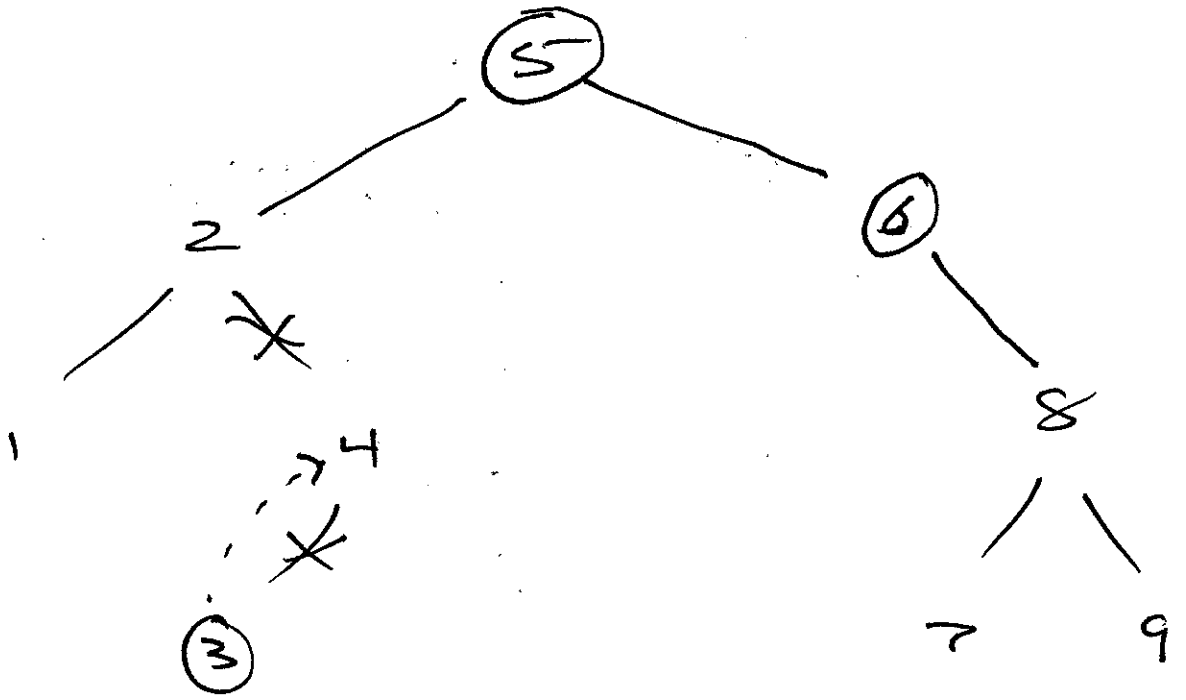
Ex.




result:



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




delet: 4, 5

result: