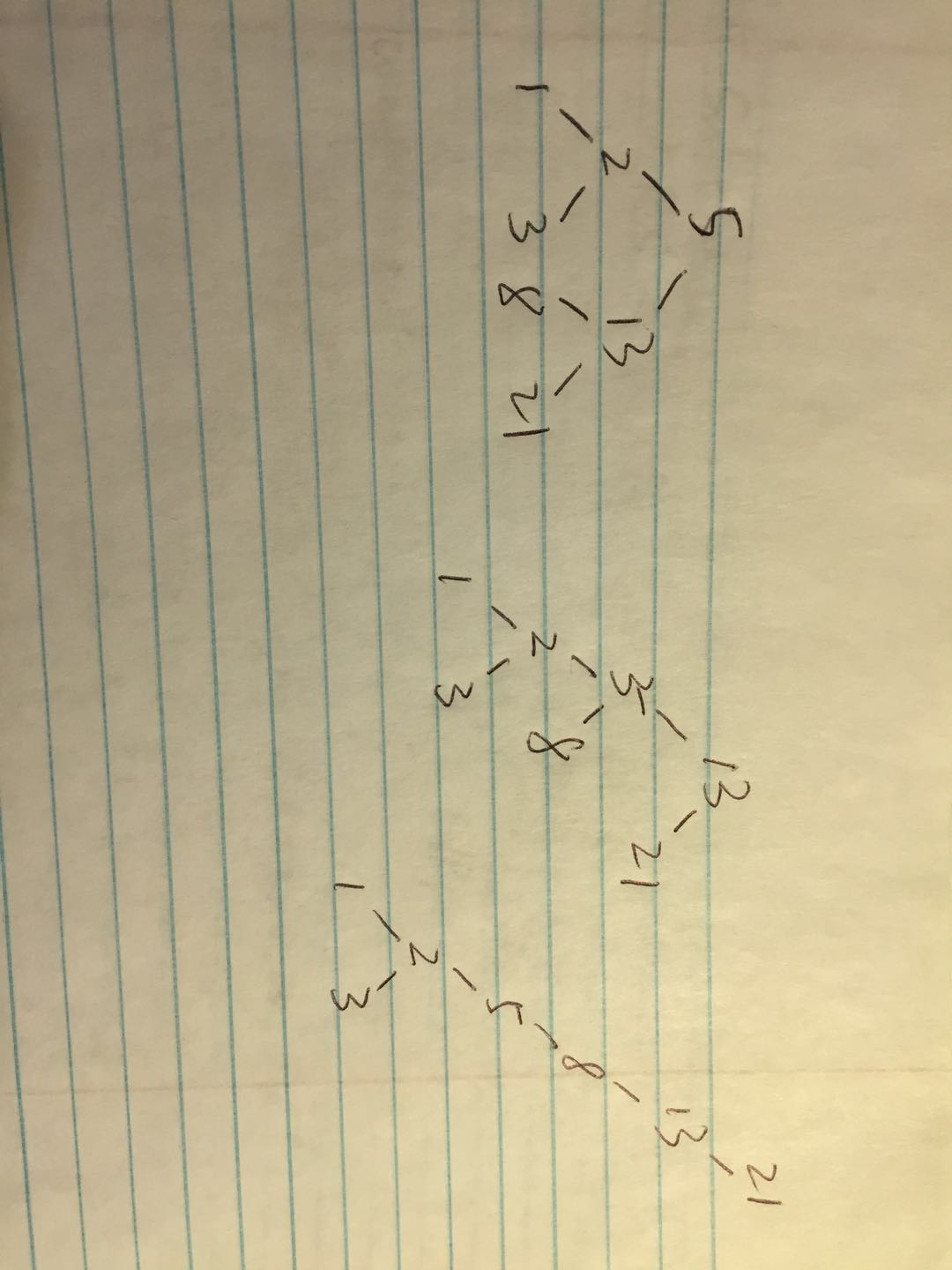
**CS 3050 Homework # 3.                                   Name : Zhiqian Zhou**

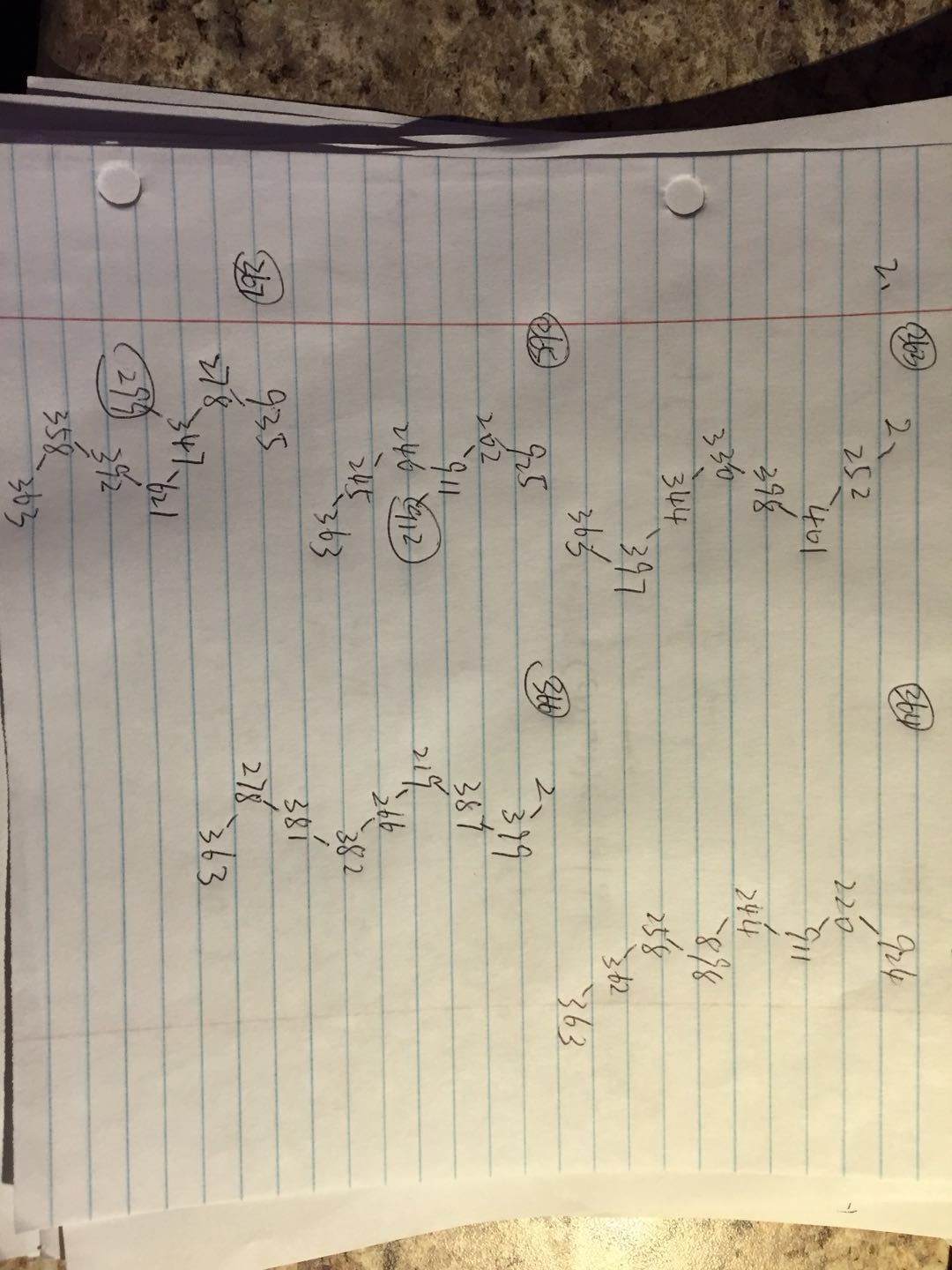
**Submitted to Canvas, due at 11:59pm on March 13, 2018.**

1. For the set A = {1, 2, 3, 5, 8, 13, 21} of keys, draw three binary search trees of different heights.



1. Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined? Explain why.
2. 2, 252, 401, 398, 330, 344, 397, 363.
3. 924, 220, 911, 244, 898, 258, 362, 363.
4. 925, 202, 911, 240, 912, 245, 363.
5. 2, 399, 387, 219, 266, 382, 381, 278, 363.
6. 935, 278, 347, 621, 299, 392, 358, 363.

365 and 367 could not be the sequence of nodes examined.



1. For a binary search tree T of n nodes, determine the number of internal nodes in O(n) running time in psuedo code.

(1) Give a recursive algorithm.

(2) Give a non-recursive algorithm.

Leaf(T)

if left[T] == NULL && right[T] == NULL

return true

return false

Recursive:

Internal(T,k)

if T == NULL || Leaf(T)

return k

k++

if left(T) != NULL && !Leaf(left(T))

Internal(left[T],k)

if right(T) != NULL && !Leaf(right(T))

Internal(right[T],k)

return k

Non-recursive:

Internal(T)

if T == NULL || Leaf(T)

return 0

Stack<TreeNode\*> s

TreeNode\* ptr = T

int count = 0

While ptr != NULL || !s.empty()

While ptr != NULL

s.push(ptr)

ptr = left(ptr)

if !s.empty()

ptr = s.top()

s.pop()

if !Leaf(ptr)

count++

ptr = right(ptr)

return count

1. Given a Binary Search Tree T and two integers a and b on the tree, print all elements in T between a and b in psuedo code.

PrintRange(T,a,b)

if T == NULL

return

Stack<TreeNode\*> s

TreeNode\* ptr = T

While ptr != NULL || !s.empty()

While ptr != NULL

s.push(ptr)

if key[ptr] < a

break

ptr = left(ptr)

if !s.empty()

ptr = s.top()

if key[ptr] > b

return

if key[ptr] >= a

printf(“%d ”,&key[ptr])

s.pop()

ptr = right(ptr)

1. Suppose that instead of each node x keeping the attribute x:p, pointing to x’s parent, it keeps x:succ, pointing to x’s successor. Give pseudocode for SEARCH, INSERT, and DELETE on a binary search tree T using this representation. These procedures should operate in time O(h), where h is the height of the tree T . (Hint: You may wish to implement a subroutine that returns the parent of a node. This is the same question of 12.3-5 in the textbook.)

PARENT(z,x)

if z == NULL

z = T.root

else

z = z.left

while(right(z) != x)

z = right(z)

return z

TREE-MAXIMUM(x)

TreeNode \*ptr = x;

while(right(ptr) != NULL)

ptr = right(ptr)

return ptr

Parent(x,k)

if right(x) == NULL

if x == left(x:succ)

x:p = x:succ

else

z = x:succ

x:p = PARENT(z,x)

else

y = TREE-MAXIMUM(x.right)

if x == left(y:succ)

x:p = y:succ

else

z = y:succ

x:p = PARENT(z,x)

return x:p

SEARCH(T,k)

While T != NULL

if key[T] < k

T = right(T)

elseif key[T] == k

return T

else

T = left(T)

return T

INSERT(T,k)

While T != NULL

par = T

if key[T] < k

succ = T

T = right(T)

else

pred = T

T = left(T)

if T == NULL

T = new TreeNode(k)

else

if k < key[par]

left(par) = new TreeNode(k)

insert = left(par)

else

right(par) = new TreeNode(k)

insert = right(par)

insert:succ = succ

if pred != NULL

pred:succ = insert

TRANSPLANT(T,u,v)

if PARENT(u) == NULL

T.root = v

elseif u == left(PARENT(u))

left(PARENT(u)) = v

else

right(PARENT(u)) = v

TREE-MINIMUM(x)

TreeNode \*ptr = x;

while(left(ptr) != NULL)

ptr = left(ptr)

return ptr

Predessor(x)

if left(x) != NULL

return TREE-MAXIMUM(left(x))

y = PARENT(x)

while y != NULL and x = left(y)

x = y

y = PARENT(y)

return y

DELETE(T,k)

pred = Predessor(T)

target = SEARCH(T,k)

if pred != NULL

pred:succ = traget:succ

if left(target) == NULL && right(target) == NULL

delete target

elseif left(target) == NULL

TRANSPLANT(T,target,right(target))

elseif right(target) == NULL

TRANSPLANT(T,target,left(target))

else

y = target:succ

if(PARENT(y) != target)

TRANSPLANT(T,y,right(r))

right(y) = right(target)

TRANSPLANT(T,target,y)

left(y) = left(target)