CS 2420 Algorithms and Data Structures

Fall Semester, 2021

Midterm Exam

3:00 p.m. – 4:15 p.m., Thursday, October 21, 2021

Your name and A-number:

Benjamin Swenson

A02299890

Total points: 100 (seven pages)

Multiple Choices (20 questions; 3 points each; 60 points in total). Please pick the single best answer.

1. For the following statements on big-O notation, which one is NOT correct?

   (a) $n^2 =$ O(n log n)
   (b) n log n = O(n log n + 4000 n)
   (c) $2n \log n + 5n = O(n^2 - 12n)$
   (d) $n^{100} = O(2^n)$

2. For the following two functions: f(n) = $n\sqrt{n}$ and g(n) = $n\ log^3 n$, which of the following is correct?

   (a) f(n) = O(g(n)) (b) f(n) = Ω(g(n)) (c) f(n) = θ(g(n)) (d) none of the above

3. What is the time complexity of the following piece of code?

```
sum = 0;
for (i = 0; i < n + 10; i++)          - n
    for (j = 0; j < i; j++)           - n
        for (k = 0; k < 1000; k++)    - 1000
            sum++;
```

   $\Rightarrow O(n^2 \cdot 1000)$

   $\Rightarrow O(n^2)$

   (a) O(n) (b) O(n^2) (c) O(n^3) (d) O(n^4)

4. What is the time complexity of the following piece of code?

```
sum = 0;
for (i = 0; i < 5n; i++)
    for (k = n/3; k < 2n/3; k++)
        sum++;
```

   (a) O(n) (b) O(n^2) (c) O(n^3) (d) O(n^4)

5. What is the time complexity of the following recursive function?

```
void fun(int n)
{
   if (n <=1)
        return;
   fun(n-1);
   for (int i = 0; i < n^2; i++)
      print "*";
}
```

(a) O(n) (b) $O(n^2)$ (c) $O(n^3)$ (d) $O(n^4)$

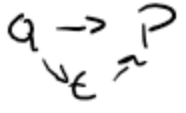
6. Consider the following recursive function. How many asterisks are printed by the code when n = 16?

```
void   doit (int n)
{ if (n <=1)
      return;
  print "*";
  doit(n/2);
}
```

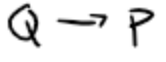
(a) 4 (b) 5 (c) 6 (d) 3 (e) none of the above

7. Suppose q is a node in a singly linked list and p is the next node of q (i.e., p = q.next). If we want to insert a new node t between q and p, which of the following operations is correct?

(a) t.next = p.next; p.next = t;
(b) p.next = t; t.next = q;
(c) p.next = t.next; t.next = p;
(d) q.next = t; t.next = p;



8. Suppose q is a node in a singly linked list and p is the next node of q (i.e., p = q.next). If we want to delete the node p, which of the following operations is correct?



(a) p.next = null;
(b) q.next = null;
(c) q.next = p.next;
(d) p.next = q.next;

9. The push and pop operations of a stack S always happen at:

(a) the top of S (b) the bottom of S (c) an arbitrary place of S (d) none of the above

10. The enqueue operations of a queue Q always happen at:

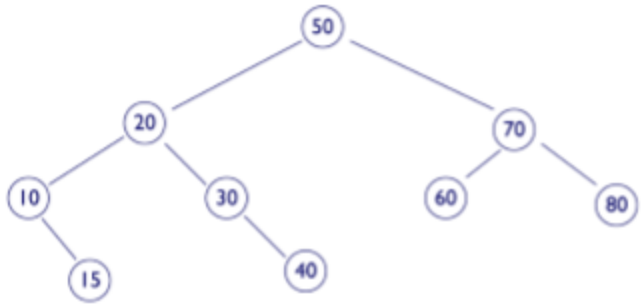
(a) the front of Q (b) the rear of Q (c) an arbitrary place of Q (d) none of the above

11. Which data structure in the following is NOT a linear structure?

(a) arrays (b) linked lists (c) AVL trees (d) queues (e) stacks

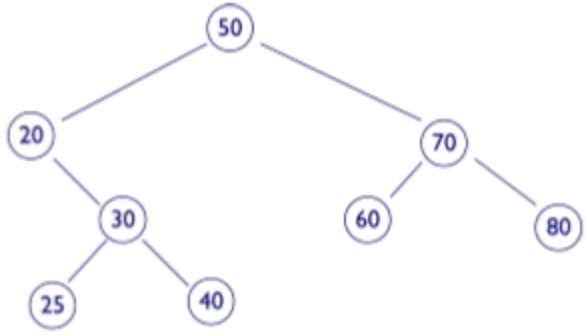
12. Consider the following binary search tree. Suppose we want to remove 20. Who will become the left child of 50?

(a) 10 (b) 20 (c) 30 (d) 40

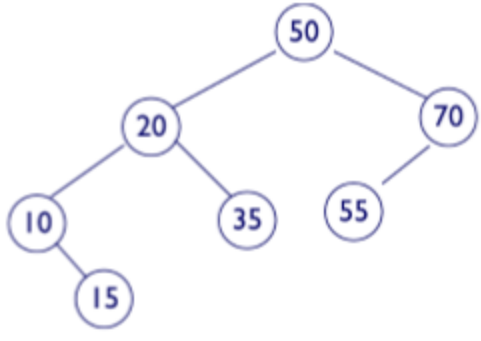


13. Consider the following binary search tree. Suppose we want to remove 20. Who will become the left child of 50?

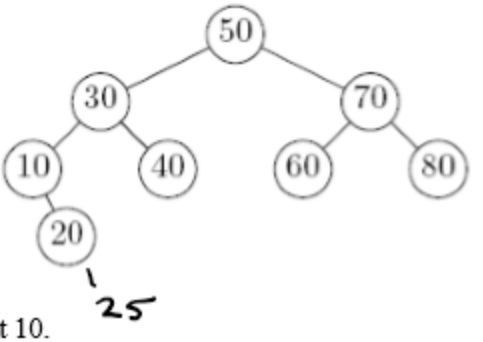
(a) 20 (b) 25 (c) 30 (d) 40



14. Is the following tree an AVL tree? (a) Yes (b) No

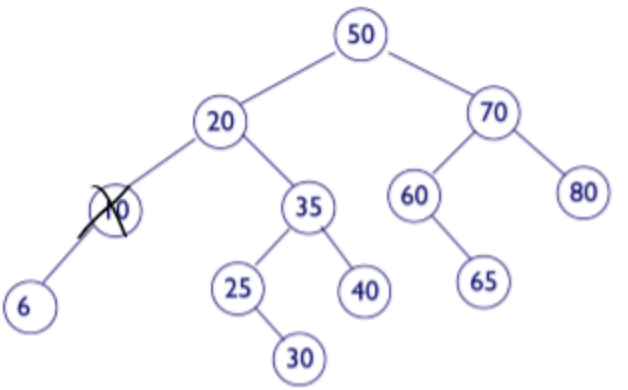


15. Suppose that we insert 25 into the AVL tree below. What rotation would be used to fix the balance, according to the algorithm we discussed in class?



(a) A double rotation rooted at 10.
(b) A double rotation rooted at 30.
(c) A double rotation rooted at 50.
(d) A single rotation rooted at 10.
(e) A single rotation rooted at 30.
(f) No rotations are required.

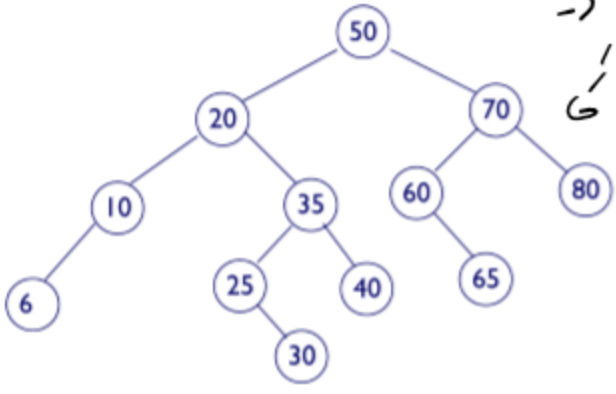
16. Remove 10 from the AVL tree below. What rotation will be required to restore the balance?



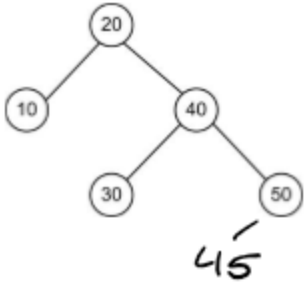
(a) A single rotation rooted at 20.
(b) A single rotation rooted at 50.
(c) A double rotation rooted at 20.
(d) A double rotation rooted at 35.
(e) A single rotation rooted at 50.
(f) No rotations are required.

17. Consider the following AVL tree. Suppose we remove 50. After the remove operation, which node will become the root of the new tree?

(a) 60 (b) 20 (c) 35 (d) 25 (e) 70



18. Insert 45 into the AVL tree below and rebalance. What is the preorder traversal of the resulting tree?



a. 20 10 45 40 30 50
b. 20 10 50 40 30 45
c. 30 20 10 40 50 45
d. 40 20 10 30 50 45
e. 40 20 10 30 45 50
f. none of the above

19. For each insert operation in an AVL tree of n nodes, what is the maximum number of rotations do we need to rebalance the tree in the worst case? (A double rotation is counted as one rotation.)

(a) 0 (b) 1 (c) O(log n) (e) none of the above

20. For each remove operation in an AVL tree of n nodes, what is the maximum number of rotations do we need to rebalance the tree in the worst case? (A double rotation is counted as one rotation.)

(a) 0 (b) 1 (c) 2 (d) O(log n) (e) none of the above

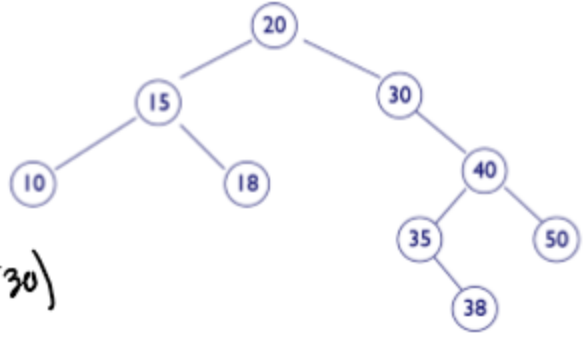
Short Answers (four questions; 10 points each; 40 points in total).

1. Suppose we have a linked list L in which each node is defined by the class below. You are given the head of the list. Write a function sum(ListNode head) that returns the sum of all values of the nodes of L (if the list is empty, then return 0). Your algorithm should run in O(n) time, where n is the number of nodes in L.

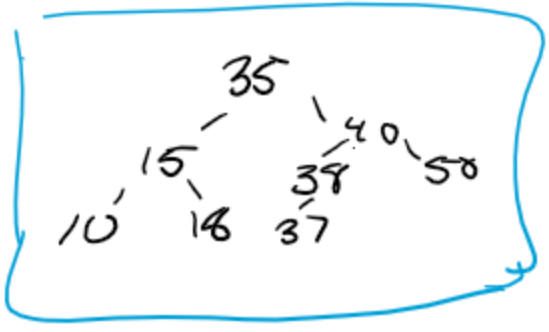
```
class ListNode{
        public int value;
        public ListNode next;
}
//given the head of the list, the following function should return the sum of all values of the nodes of L
int sum(ListNode head) {
  int sum = 0
While head.next != null
    Sum += head.value
    head = head.next

Sum += head
return (sum)
```

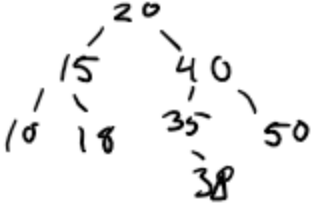
2. Consider the following binary search tree (BST). Draw the new tree after the following operations in order: remove(30), insert(37), remove(20). This is an ordinary BST, so you do not need to perform any rotations. **Note:** If you need to remove a node v that has two children, then you are required to use the smallest node in the right subtree of v to replace v.



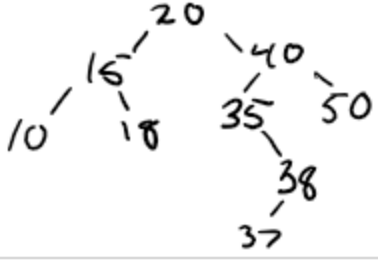
rmv(20)



rmv(30)



insert(37)



3. Suppose we have a binary search tree T in which each node is defined in the following class. For each node x of T, recall that the **depth** of x is defined to be the number of edges in the path of T from the root to x. For example, consider the tree in Question 4; the depth of 90 is 2, the depth of 60 is 3, the depth of 50 is 0, etc. Given the root of T and a key x, write a function depth(TreeNode v, int x) so that when we call depth(root, x), it returns the depth of x if x is in T, and returns -1 otherwise. Your algorithm should run in O(h) time, where h is the height of T.

```
class TreeNode{
        public int key;
        public TreeNode left, right;
}
```

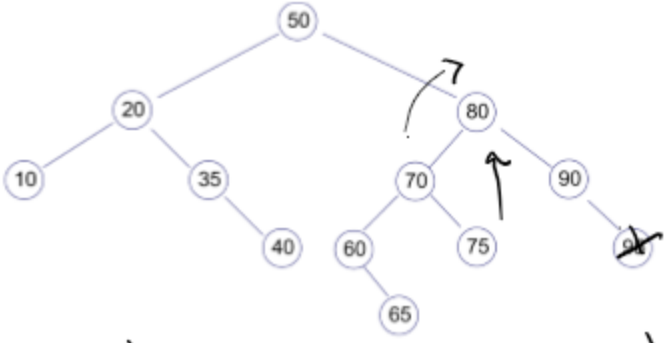
I'm gonna try recursion

```
int depth(TreeNode v, int x) {
int depth = 0
if v = null:
    return -1
if v.key = x:
   return depth
if v.key > x:
   depth = depth(key.left, x)
    depth ++
else:
   depth = depth(key.right, x)
   depth ++
```

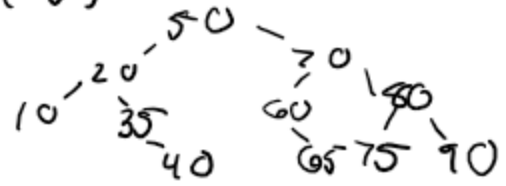
at the end out of all statements

return depth

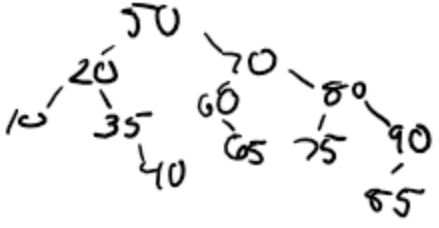
4. Consider the following AVL tree. Draw the new AVL tree after we perform the following operations in order: remove(95), insert(85), remove(10).



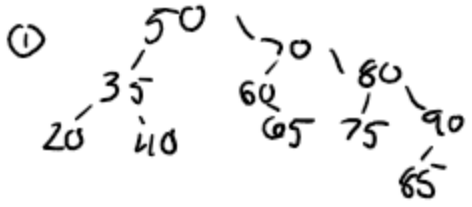
rmv(95)



insert(85)



rmv(10)

① 

②