Name: _____Key_____
Email address: _____

CSE 373 Sample Midterm #1

(closed book, closed notes, calculators o.k.)

Instructions Read the directions for each question carefully before answering. We may give partial credit based on the work you **write down**, so if time permits, show your work! Use only the data structures and algorithms we have discussed in class or which were mentioned in the book so far.

Note: For questions where you are drawing pictures, please circle your final answer for any credit. There is one extra page at the end of the exam that you may use for extra space on any problem. If you detach this page it must still be turned in with your exam when you leave.

Advice You have 50 minutes, **do the easy questions first**, and work quickly!

1. (16 pts) Big-Oh and Run Time Analysis:

a. (10 points total) Describe the running time of the following pseudocode in Big-Oh notation in terms of the variable n. Assume all variables used have been declared. ***Show your work for partial credit.***

```
int foo(int k) {
      int cost;
      for (int i = 0; i < k; ++i)
            cost = cost + (i * k);
      return cost;
}

I. answ = foo(n);
```

O (n)

```
II.   int sum;
      for (int i = 0; i < n; ++i)
      {
            if (n < 1000)
                  sum++
            else
                  sum += foo(n);
      }
```

O (n^2)

```
III.  for (int i = 0; i < n + 100; ++i)
      {
            for (int j = 0; j < i * n ; ++j)
            {
                  sum = sum + j;
            }

            for (int k = 0; k < n + n + n; ++k)
            {
                  c[k] = c[k] + sum;
            }
      }
```

O (n^3)

```
IV.   for (int j = 4; j < n; j=j+2) {
            val = 0;
            for (int i = 0; i < j; ++i)
            {
                  val = val + i * j;
                  for (int k = 0; k < n; ++k)
                  {
                        val++;
                  }
            }
      }
```

O (n^3)

```
V.    for (int i = 0; i < n * 1000; ++i)
      {
            sum = (sum * sum)/(n * i);
            for (int j = 0; j < i; ++j)
            {
                  sum += j * i;
            }
      }
```

O (n^2)

1. (cont.) Big-Oh and Run Time Analysis

b. (6 pts) Consider the following function:

```
int mystery(int n)
{
     int answer;
     if (n > 0)
     {
          answer =(mystery(n-2)+3*mystery(n/2) + 5);
          return answer;
     }
     else
          return 1;
}
```

Write down the complete recurrence relation, T(n), for the running time of `mystery(n)`. Be sure you include a base case T(0). **You do not have to actually solve this relation, just write it down.**

T (n) = T (n-2) + T (n/2) + C, for N > 0
T (n) = 1, for N = 0

2. (9 points) Trees – (3 points each)

a) What is the minimum and maximum number of nodes in a *complete* binary tree of height h?

min = 2^{h}
max = $2^{h+1} - 1$

b) What is the minimum and maximum number of nodes in a *perfect* binary tree of height h?

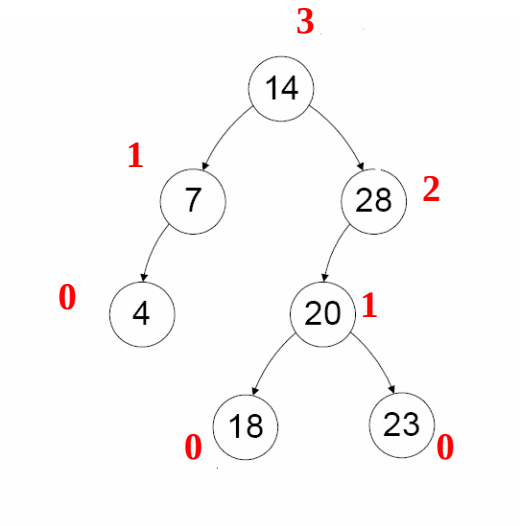
min = $2^{h+1} - 1$
max = $2^{h+1} - 1$

c) What is the AVL balance property?

At every node, the height of its left subtree differs from the height of its right subtree by no more than 1.

3. (20 pts)

a.) (10 pts) Mark the height for each node of the tree below.



b.) (10 pts) Also, circle **yes** or **no** to indicate whether the tree above might represent each of the following data structures. If you circle **no,** give **one specific reason** why the tree could **not** be that data structure.

• Binary tree **yes** no
Each node has at most 2 children.

• Binary search tree **yes** no
Every node has at most two children, so it's a binary tree.

• AVL tree yes **no**
Balance is broken under node 28.

• Splay tree **yes** no
Any binary search tree is a splay tree. This tree has the search tree property; so, it's a splay tree.

4. (16 pts) Running Time Analysis

Give an O-bound on the *worst case* running time for each of the following in terms of *n.* **No explanation is required**, but an explanation may help for partial credit. Assume that all keys are distinct.

(a) insert in a *Binary Search Tree* of size *n*

worst case*: O(n) – since the tree is unbalanced. For instance, the tree could be all left children, and then an insert of a new minimum element would have to traverse down all N children.*

(b) insert in an *AVL tree* of size *n*

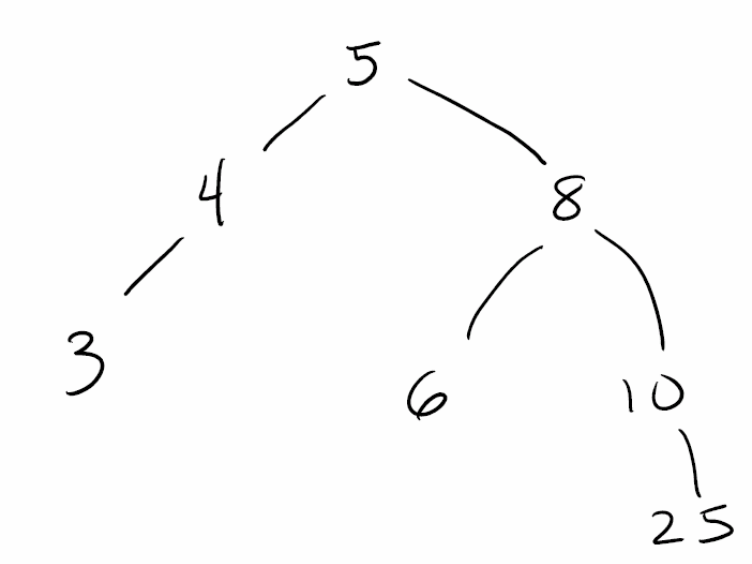
worst case: O(log n)
AVL trees are balanced, and a new element goes at the fringe, so this must take take logarithmic time (best, average, worst, or any other case). Some of you suggested that we might get lucky and find a NULL child high in the tree where the new element should go. However, if such a NULL child existed, its parent would be out of balance (one child has large height, the other has height of –1).

(c) insert in a *splay tree* of size *n*

worst case*: O(n) – since the tree is unbalanced. For instance, the tree could be all left children, and then an insert of a new minimum element would have to traverse down all N children, then splay back up. Note that the amortized guarantee of a splay tree only applies to a sequence of operations – doesn't prevent individual bad cases like this.*

5. (20 pts)

a. (8 pts) Draw the AVL tree that results from inserting the keys 4, 10, 3, 8, 5, 6, 25 in that order into an initially empty AVL tree. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.***



Insertion of 5 causes imbalance at node 10 – single rotation makes 8 the right child of the root (4).

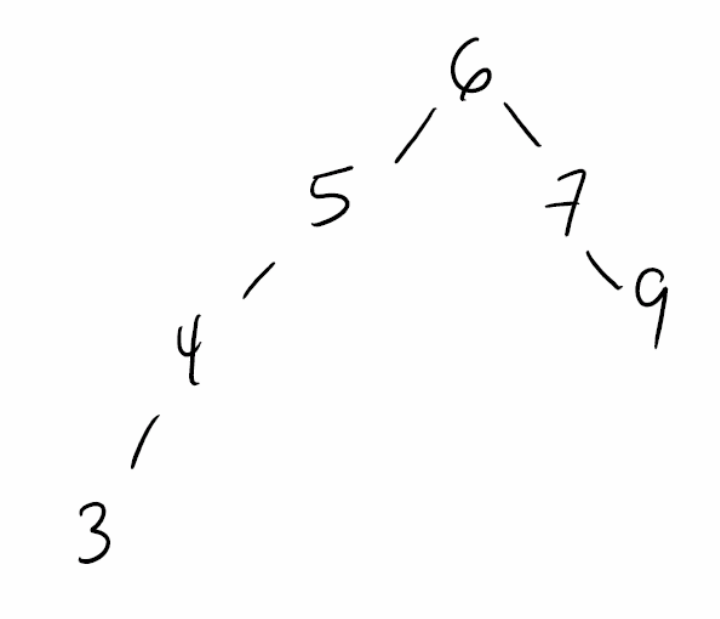
Insertion of 6 causes imbalance at root (4) – double rotation needed, brings 5 to the root.

Final tree is balanced.

b. (2 pts) Give a *preorder* traversal of your final AVL tree you created in part a.

5, 4, 3, 8, 6, 10, 25

c. (8 pts) Draw the Splay tree that results from inserting the keys 4, 9, 3, 7, 5, 6 in that order into an initially empty Splay tree. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.*** You may continue your answer to this problem on the next page if needed.

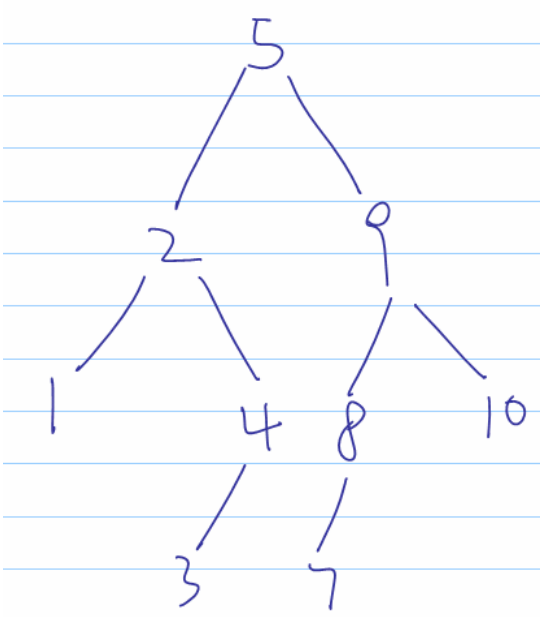


d. (2 pts) Give a *postorder* traversal of your final Splay tree you created in part c.

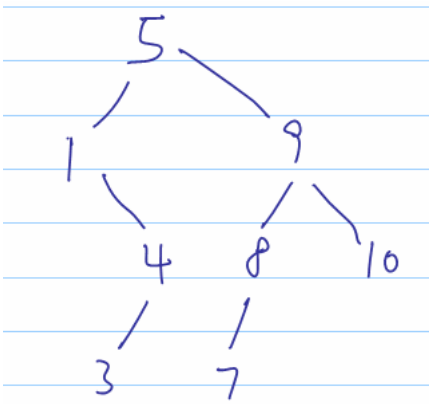
3, 4, 5, 9, 7, 6

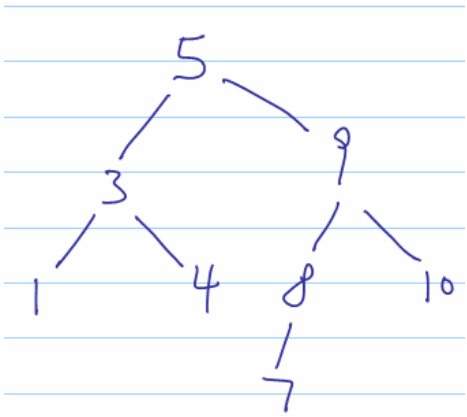
6. Binary search trees (9 pts)

a) (5 points) Insert the following values into a binary search tree in this order: 5, 9, 2, 1, 4, 8, 3, 7, 10. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.***



b) (4 points) Show the tree you created above after deleting the value 2 using one of the (two) methods described in lecture. (do not use lazy deletion)





7. More Big-O – True or false (10 pts, 2 pts each)

- $3000N + 36 = O(N)$ T
- $N \log N + N/5 = O(N^2)$ T
- $N \log N + N^2 = O(N \log N)$ F
- $N^2 \log N + N^2 = O(N^2)$ F
- $N^{1/2} + \log N = O(\log N)$ F