1) 10 Points

Compute an appropriately tight O (big-O) bound on the running time of each code fragment, in terms of *n*. Assume integer arithmetic. Circle your answer for each fragment.

a)
```
for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
           for(k = 0; k < i * j; k++) {
                sum++;
           }
      }
}
```

n^2

$O(n^4)$

b)
```
for(i = 1; i < n; i = i * 2) {
      for(j = 1; j < i; j++) {
           sum++;
      }
}
```

log n

n

$O(n \log n)$

Actually, this is $O(n)$

It executes $2n-1$ increments

c)
```
for(i = 0; i < n; i++) {
      myArray = new array[i];
      for(j = 0; j < i; j++) {
           myArray[j] = random();
      }
      mergeSort(myArray);
}
```

n

n

n log n

$O(n^2 \log n)$

d)
```
for(i = 0; i < n; i++) {
      tree = new UnbalancedBinarySearchTree();
      for(j = 0; j < n; j++) {
           tree.insert(j);
      }
}
```

n

n

n

"list" tree results

$O(n^3)$

e)
```
tree = new AVLTree();
for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
           tree.insert(random());
      }
}
```

n

n

$O(n^2 \log n)$

inserts n^2 items at log cost for each

Q3: O, Ω, and Θ (9 pts)

For each of the following statements, indicate whether it is always true, sometimes true, or never true. You do not need to include an explanation. Assume that the domain and codomain of all functions in this problem are natural numbers(1, 2, 3 ...).

a) A function that is $O(n)$ is ________ $O(n^2)$.

● Always ○ Never ○ Sometimes

b) A function that is $\Theta(n)$ is ________ $O(n^2)$

● Always ○ Never ○ Sometimes

c) A function that is $O(n)$ is ________ $O(\log n)$.

○ Always ○ Never ● Sometimes

d) A function that is $O(n)$ is ________ $\Theta(n^2)$

○ Always ● Never ○ Sometimes

e) A function that is $\Omega(\log(n^{1/\log n}))$ is ________ $\Omega(n^{2/3})$.

○ Always ○ Never ● Sometimes

f) A function that is $\Omega(\log n)$ is ________ $\Omega(n)$

○ Always ○ Never ● Sometimes

g) A function that is $O(n)$ is ________ $\Omega(n^2)$.

○ Always ● Never ○ Sometimes

h) If $f(n)$ is $\Theta(g(n))$ and $g(n)$ is $\Omega(h(n))$, then $f(n)$ is $\Omega(h(n))$.

● Always ○ Never ○ Sometimes

i) If $f(n)$ is both $O(g(n))$ and $\Omega(h(n))$, then **g(n)** is $\Theta(h(n))$.

○ Always ○ Never ● Sometimes

Q4: Write a Recurrence (5 pts)

Give a base case and a recurrence for the runtime of the following function. Use variables appropriately for constants (e.g. c_1, c_2, etc.) in your recurrence (you do not need to attempt to count the exact number of operations). **YOU DO NOT NEED TO SOLVE** this recurrence.

```
public static int fun(int n) {
    if (n < 5) {
        return n;
    } else if (fun(n / 2) < 10) {
        int a = fun(n - 3);
        int b = fun(n / 2);
        return b;
    } else {
        int j = fun(n - 7);
        for (int i = 0; i < n * n; i += 4) {
            j += i;
        }
        return j;
    }
}
```

$T(n) =$ ______________c_0______________________ For $n < 5$

$T(n) =$ __$2T(n/2) + T(n-3) + c_1$__ For $n > 5$

Yipee!!!! YOU DO NOT NEED TO SOLVE *this* recurrence…

Note that the function always returns a value that is less than 5.

(by induction)
Base case: Clearly true for n < 5.
IH: fun(i) < 5 for all i < k
Inductive step: show fun(k) < 5.
The else-if branch calculates fun(k/2), which is less than 5 because k/2 < k implies fun(k/2) < 5.
Thus fun(k) enters this branch and returns b = fun(k/2) < 5, so fun(k) < 5.
By induction, fun(k) < 5 for all n.

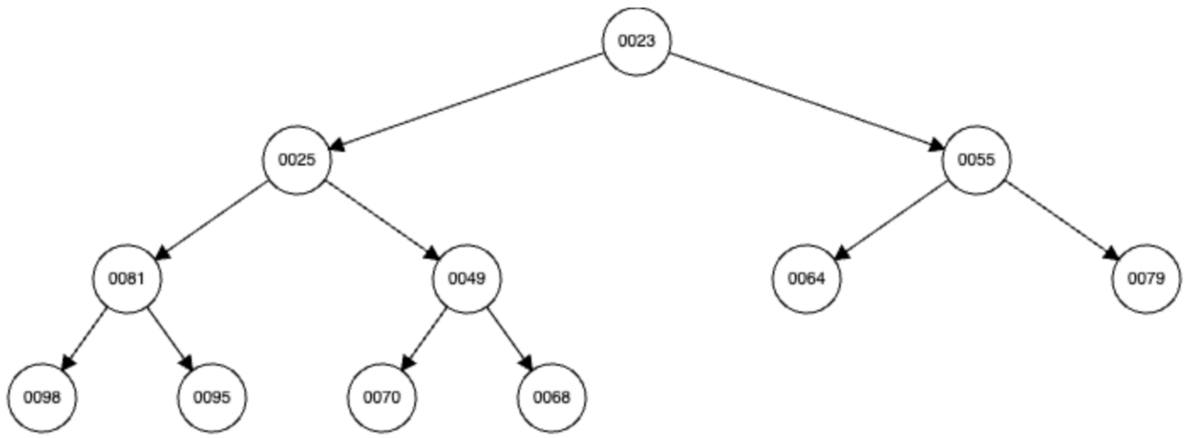
Q6: Heaps (11 pts)

Here's an array presentation of a 0-indexed binary heap:

23	25	55	81	49	64	79	98	95	70	68		

a) (1 pt) This is a ___min____ (choose one: max/min) heap.

b) (2 pts) Draw the visual representation of the given heap.



c) (2 pts each) For each statement below, if there's an **insertion** that can achieve the goal, list the element to insert; otherwise, justify why it cannot be achieved (1-2 sentences max). Each statement is independent; you are allowed to insert one element for each statement.

i) 70 is a child of 64

Insert(70)

ii) 79 is no longer a child of 55

Insert any number < 55

iii) 49 only has one child

Impossible. 49 already has two children, and we are performing only one insertion, which will add a child to 64. This will not impact 49. (Mentioning maintaining the heap property is also correct.etc)

iv) No percolation is done for the insertion

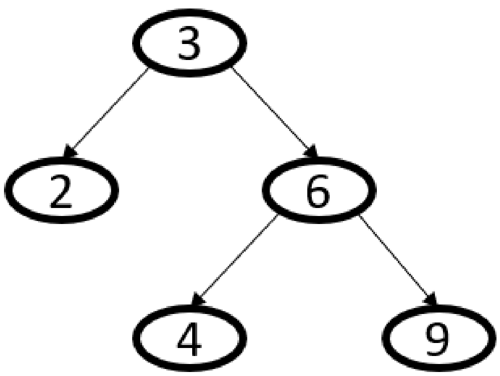
Insert any number >= 64

Q7: AVL (12 pts)

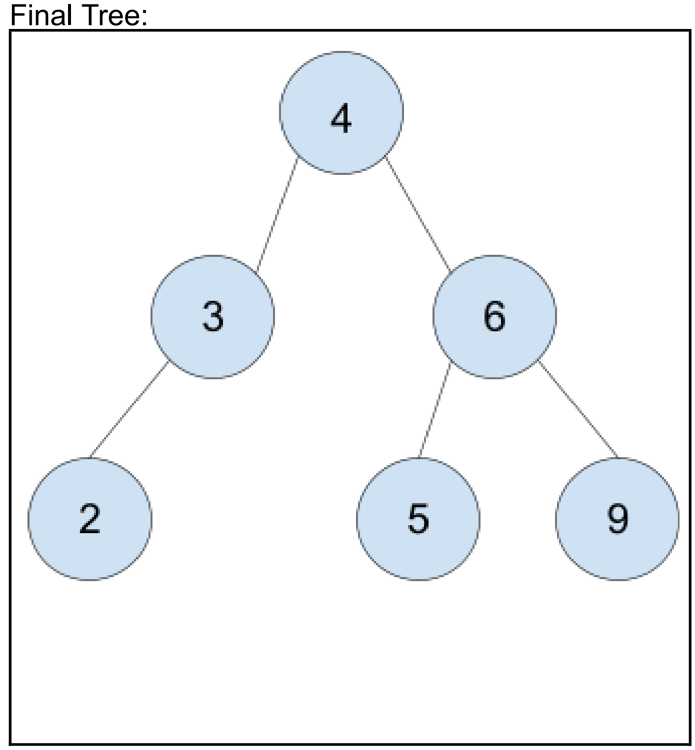
a) (3 pts) Draw the AVL tree that results after **inserting 5** into this AVL tree. Be sure to draw your final tree in the box below AND **indicate the total number of single and double rotations required for this insertion.**

b) (1 pts) This insertion required: (SELECT ONE)

- ○ One single rotation
- ● One double rotation
- ○ One single rotation AND One double rotation
- ○ More than one single rotation and one double rotation



Final Tree:



Intermediate Tree: