**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++;
         }
     }
 }
```

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

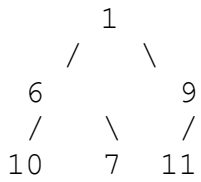
c)
```
for(i = 0; i < n; i++) {
     myArray = new array[i];
     for(j = 0; j < i; j++) {
         myArray[j] = random();
     }
     mergeSort(myArray);
 }
```

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

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

**3)10 Points**

Consider this binary min-heap:




Perform the following operations in order, drawing the result after each operation and using it as the starting point for the next operation. You only need to show the result of the operation, but showing your work will allow partial credit in case of error.
If the space here is insufficient, use the back of this sheet (clearly labeling your work).
Circle the result of each operation so we can distinguish it from intermediate work.

a) DeleteMin

b) Insert 8

c) Insert 2

d) Draw an efficient array-based representation of your final heap from step *c*.

e) In your array-based representation, what is the index of:

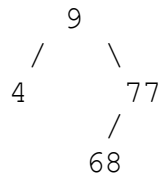
the parent of the node at index *i*:

the left child of the node at index *i*:

the right child of the node at index *i*:

**4) 10 Points**

Consider this AVL tree:




Perform the following operations in order, drawing the result after each operation and using it as the starting point for the next operation. You only need to show the result of the operation, but showing your work will allow partial credit in case of error.
Circle the result of each operation so we can distinguish it from intermediate work.

a. Insert 52

*b.* Delete 9 *We did not cover general AVL delete. But if you did the previous operation correctly, you can delete this without creating an imbalance.*

c. Insert 75

d. Insert 55