4. The-ta Knows Best! [6 points]
For each of the following, give a $\Theta(-)$ bound, in terms of n, for the *worst case runtime* of the method.

(a) (2 points)

```
1  int hello(int n) {
2     if (n == 0) {
3        return 0;
4     }
5     for (int i = 0; i < n; i++) {
6        for (int j = 0; j < n * n; j++) {
7           System.out.println("HELLO");
8        }
9     }
10    return hello(n - 1);
11 }
```

Runtime

(b) (2 points)

```
1  void whee(int n) {
2     for (int i = 1; i < n; i *= 2) {
3        for (int j = 1; j < n; j *= 3) {
4           System.out.println("WHEE!");
5        }
6     }
7     for (int k = n/2; k < n; k++) {
8           System.out.println("WOAH!");
9     }
10 }
```

Runtime

(c) (2 points)

```
1  void flipflop(int n, int sum) {
2     if (n > 10000) {
3        for (int i = 0; i < n * n * n; i++) {
4           sum++;
5        }
6     }
7     else {
8        for (int i = 0; i < n * n * n * n; i++) {
9           sum++;
10       }
11    }
12 }
```

Runtime

7 Writing A Recurrence

[6 points] Write a recurrence describing the running time of the recursive code below. Use constants like c_1 and c_2 when describing the number of operations: you do not have to give an exact count.

You do not need to solve for a closed form of THIS recurrence

```
int foobar(int n){
     if(n <= 15){
          return 2n + 8
     }
     else{
          for(int i = 0; i < n; i++){
               for(int j = 0; j < i; j++){
                    print i
               }
          }
          return foobar(n/2) * foobar(n-3) + n
     }
}
```

$$T(n) = \begin{cases} \\ \\ \\ \end{cases}$$

Don't find a closed form of *this* recurrence.

8 Solving A Recurrence

[15 points] In this problem you will find a closed form of the following recurrence, and check your answer with the Master Theorem.

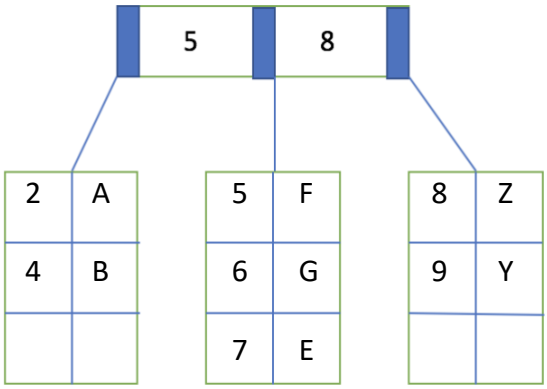
$$T(n) = \begin{cases} 3 & \text{if } n = 1 \\ 4T(n/2) + 2n^2 & \text{otherwise} \end{cases}$$

1. Solve the recurrence. Your final answer should not have any recursion or summations, but need not "look nice." You should not simplify to a big-Θ answer in this part, keep all the exact constants in your answer. You may use either unrolling or the tree method. In either case, you must show your work for **ANY** credit. The list of steps for the tree method and the list of summations on the last page may come in handy.

2. Use the Master Theorem (see the last page) to get a big-Θ bound on the recurrence.

B-Trees

Below is a B-Tree storing integer keys with character values. Insert the (key, value) pairs (3, F) and (1, W) in that order into the B-Tree below. You should draw two B-Trees: one for after (3, F) was inserted and one after both were inserted.



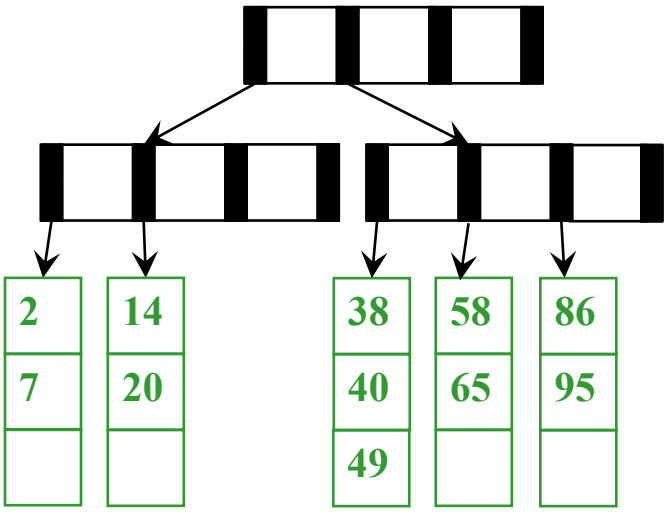
9. (9 pts) B-trees

a) (1 pt) In the **ORIGINAL** B-Tree shown below, **add values for the interior nodes**.

b) (4 pts) Starting with the **ORIGINAL** B-tree shown below, in upper box, draw the tree resulting after inserting the value 50 (*including values for interior nodes*). Use the method for insertion described in lecture and in the book.

c) (4 pts) Starting with the **ORIGINAL** B-tree shown below, in the lower box, draw the tree resulting after deleting the value 14 (*including values for interior nodes*). Use the method for deletion described in lecture and in the book.

ORIGINAL:



After inserting 50:

After deleting 14: