3. (12 points) Can you handle the truth?
Fill out the following table with boolean values of true/false.

| A    | b    | !a & b | a || (a & b) | (a || b) && (!a || lb) | !(a && b) || (a && b) |
|------|------|--------|-------------|------------------------|------------------------|
| true | true | false  | false       | false                  | true                   |
| true | false|        | true        | true                   |                        |
| false| true | false  | false       | true                   | false                  |
| False| false|        | false       |                        |                        |

4. (15 point) Big Oh time
Give the running time of the following code segments as in terms of big-Oh of n. Also, give values of a, b, and x when they are invoked with n = 50. Assume that initially a, b, and x are defined as follows before every code segments.

```java
int a = 100;
int b = 10;
double x = 16.0;
```

// Note: Even though the values are calculated with n = 50 (a constant), please express your running time as function of n, i.e. O(f(n)), considering that it can be set to any value.

a. (2 pts) Code segment 0
   ```java
   b = a;
   if (a > 0) { a = b; }
   ```
   Running time = \(O(1)\) b = 100

b. (5 pts) Code segment 1
   ```java
   for (int i = 0; i < n/5; i+=2) {
   x = x/2;
   a = a+2;
   b++;
   }
   ```
   Running time = \(O(n)\) a = 32.00 b = 15 x = 0.5

c. (5 pts) Code segment 2
   ```java
   for (int i = 0; i < n; I = i+10) {
   for (int j = 0; j < i; j++) {
   x = ++aa + b++;
   }
   }
   ```
   Running time = \(O(n^2)\) a = 200 b = 110 x = 329

d. (3 pts) Code segment 3
   ```java
   for (int i = 0; i < n/10; i++ ) {
   for (int j = 0; j < n; j=j+2) {
   a = b++;
   }
   }
   ```
   Running time = \(O(n^2)\) Never ends so Can't be determined

e. (3 pts) In the code segments above, circle a, b, c, or d based on the rank order using big-Oh running times of the associated code segments above.
   i. Fastest b c a d
   ii. 2nd fastest a b c d
   iii. 3rd fastest a b d c
   iv. Slowest a b d c
5. **(6 points) 2-3 Trees Search:** Consider the following 2-3 tree. Show the result of add I, B, F. (You need to follow the correct insertion process for each step and show the final result).

![2-3 Trees Search Diagram]

**Anatomy of a 2-3 search tree**

6. **(6 points) Red-Black Tree:** Draw the right-leaning red-black tree equivalent to the 2-3 tree below. Indicate a red link by a double line.

![Red-Black Tree Diagram]

**Anatomy of a 2-3 search tree**

7. **(4 points) Tic-Tac-Toe:** Consider the game of 3x3 Tic-Tac-Toe. Using symmetry (rotation/flips), what is the minimum number of unique first moves you have to consider? __3__

For each first move, what is the unique number of responses (considering symmetry/rotation/flips), you have to consider? __1, 2, 5, 2, 5, 3, 7__

(1, 2, 3 don't need order)

8. **(6 points) Minimum Spanning Tree**
Using Kruskal’s or Prim’s algorithm (starting with vertex 0), number the edges in the order you will add them to the Minimum Spanning Tree in the figure on the right. Consider the weight to the edges to be:


Checkmark the algorithm you used: [ ] Kruskal’s or [ ] Prim’s?

Hint: Regardless if the algorithm you use, I marked the edge that would be labeled 1 to get you started.

![Minimum Spanning Tree Diagram]