IMPORTANT FIRST STEPS:

1. **Close your laptops and put them away.**
2. **Form a group of 2-3 students.**
3. **Clearly put your names and IDs on 1 copy of this worksheet.**
4. **Be sure to turn this exercise in at the end of class.**

Trees

Translate the following mathematical expressions into binary expression trees:

(1+2) / 4 + 6 * 9

```
       +
      /  \
     /      *
   /  \   /   \
  +   4 6     9
 / \
1   2
```

5 * 2 / 3 + (4 - 3 * 6 + 1)

```
          +
        /     \
      /         +
     /  \      /  \
    *    3    -     1
   / \      /  \
  5   2    4    *
               /  \
              3    6
```

Print the results of navigating each tree by a prefix, postfix and infix traversal:

Tree 1

Prefix: + / + 1 2 4 * 6 9

Postifx: 1 2 + 4 / 6 9 * +

Infix: 1 + 2 / 4 + 6 * 9

Tree 2

Prefix: + / * 5 2 3 + - 4 * 3 6 1

Postifx: 5 2 * 3 / 4 3 6 * - 1 + +

Infix: 5 * 2 / 3 + 4 - 3 * 6 + 1

Insert the following elements into a binary search tree:

6 4 7 10 5 2 3 14 8

```
        6
       / \
      4   7
     / \    \
    2   5    10
     \       / \
      3     8   14
```

Delete 14 and draw the resulting tree:

```
        6
       / \
      4   7
     / \    \
    2   5    10
    \        /
     3      8
```

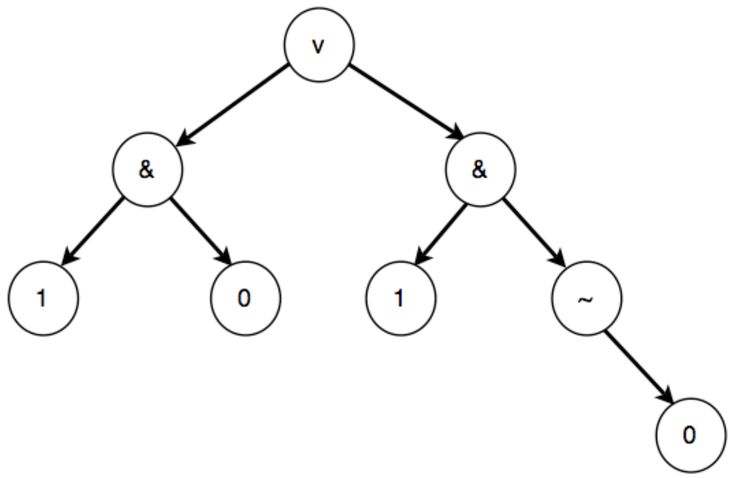
Delete 10 and draw the resulting tree:

```
        6
       / \
      4   7
     / \   \
    2   5   8
     \
      3
```

Delete 6 and draw the resulting tree:

```
        5
       / \
      4   7
     /     \
    2       8
     \
      3
```

As you know, computers are collections of logical circuits that take a set of boolean input values and produce a boolean output. You can represent this mapping from input to output with a logic expression consisting of the boolean logic operators AND (&), OR (v), and NOT (~), and the boolean values true (1) and false (0). Just as you can construct an arithmetic expression tree from an arithmetic expression, you can also construct a logic expression tree from a logic expression. Given the following logic expression tree, determine the baseline expression and whether it is true or false:



(1 ^ 0) v (1 ^ ~0)
0 v (1 ^ 1)
0 v 1
1