0. Building a Circuit

Get index cards and 3 colors of string.
Draw gates on the index cards.
Connect the gates with one color of string as the "wire." (This shows how the inputs and outputs of your gates connect.) NOTE: Wires can "branch."
Choose one of the leftover colors to be TRUE and the other FALSE. Lay colored string on top of the "wire" to simulate how the circuit performs the calculation.
You can also use the simulator at https://academo.org/demos/logic-gate-simulator/

1. Build a Half Adder

Just like in decimal arithmetic, when adding binary digits you may have to "carry the one."
Fill in the truth table for a circuit with two inputs **X** and **Y** and two outputs **Sum** and **Carry**, then build a circuit that calculates this truth table. (Hint: I used 2 XOR, OR, or AND gates)

X	**Y**	**Sum**	**Carry**

2. Build a Full Adder

When you line up numbers to perform addition, at each column after the first, you add the **Carry-In**, **X**, and **Y** to produce **Carry-Out** and **Sum**. Fill in the truth table, then build a circuit. (Hint: I used 5 XOR, OR, or AND gates that included two half-adders)

X	**Y**	**Carry-In**	**Sum**	**Carry-Out**

3. Satisfiability

Given some circuit with many inputs and one output, if there is some input of 0's and 1's to your circuit that will make the output 1 (TRUE) then the circuit is satisfiable.
In the same way, if there is an assignment of variables in a Boolean formula such that the formula evaluates to 1 (TRUE) than the formula is satisfiable.

It is NP-Complete to determine whether a formula or circuit is satisfiable.

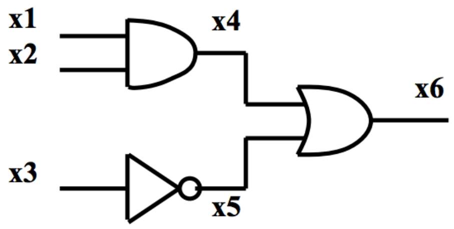
Build, draw, or simulate a circuit with at least 3 gates that is satisfiable. Then make a circuit/gate with at least 3 gates that is NOT satisfiable.

Write the Boolean formulas and truth tables for both circuits.

4. Circuits to Formulas

As you might imagine, for every circuit, there is a Boolean formula that is SATisfiable if and only if the formula is satisfiable. This is how the reduction works to prove that Circuit-SAT is NP-Complete given that Boolean-Formula-SAT is NP-Complete.
It's pretty straightforward to translate a circuit to a Boolean formula. Write out the satisfiability formulas for the two circuits you created in part (3).

Example: (x1 AND x2) OR (NOT x3)



Let's do this translation a different way. Create some new variables that correspond to the output of each gate. Then, use the operation ←→ in your formula, that is an "equals sign," to require that for each gate in the circuit the input "equals" the output.
Write the formulas in this way for the circuits from part (3).

Example (above): (x1 AND x2 ←→ x4) AND (x3 ←→ NOT x5) AND (x4 OR x5 ←→ x6)

5. The Boolean "equals sign"

←→ is not a Boolean operator. But for any gate, it can be written out in terms of ANDs, ORs, and NOTs! For a NOT gate with input **X1** and output **X2**, the expression (NOT **X1** ←→**X2**) can be written out as

(X1 AND (NOT X2)) AND (X2 AND (NOT X1))
alternatively (meaning the same thing)
(X1 OR X2) AND ((NOT X1) OR (NOT X2))

Without using ←→, write out the expressions
(**X1** OR **X2** ←→ **X3**)

(**X1** AND **X2** ←→**X3**)

Extra time: Do this for the other Boolean operators

6. Universal Gates

Any Boolean function can be created with only NAND or only NOR gates. Prove it! Make all the other gates (AND, OR, NOT, XOR, NAND/NOR) with just NAND or just NOR gates.