Exercise 24: Build a majority-rules circuit. This is a circuit that has three inputs and one output. The value of its output is 1 if and only if two or more of its inputs are 1; otherwise, the output of the circuit is 0. For example, if the three inputs are 0, 1, 1, your circuit should output a 1. If its three inputs are 0, 1, 0, it should output a 0. This circuit is frequently used in **fault-tolerant computing** - environments where a computer must keep working correctly no matter what, for example as on a deep-space vehicle where making repairs is impossible. In these conditions, we might choose to put three computers on board and have all three do every computation; if two or more systems produce the same answer, we accept it. Thus, one of the machines could fail and the system would still work properly.

Solution. The truth table for the majority-rules circuit is given below:

A	B	C	$Output$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

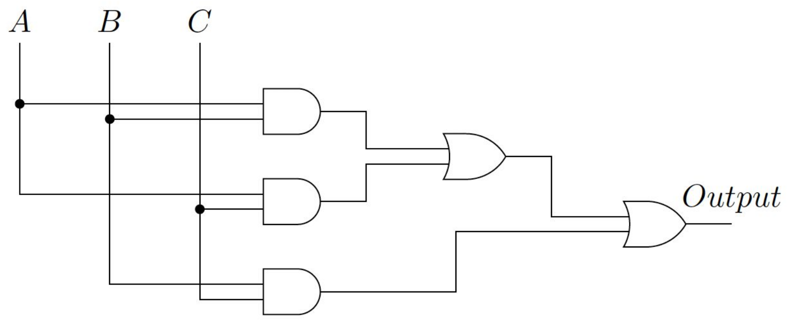
We observe that the equations to generate an output of 1 are:

$$Output = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

We can further optimize this circuit by examining the equation above. We see that for each possible combination, we require that only two of the Boolean variables are true. The value of the third negated variable is not necessary to this circuit. For example, $\overline{A}BC + ABC$ can be reduced to just *BC*. Thus, our optimized Boolean equation is

$$Output = AB + AC + BC$$

We shall use the optimized circuit for our circuit diagram.



Exercise 26: Design a 1-bit subtraction circuit. This circuit takes three inputs – two binary digits a and b and a borrow digit from the previous column. The circuit has two outputs – the difference (a b), including the borrow, and a new borrow digit that propagates to the next column. Create the truth table and build the circuit. This circuit can be used to build N-bit subtraction circuits.

Solution. N.B. This solution was prepared with the help of Sean Liu.

Review of Subtraction:

Subtraction works from right to left. At any column, we may need to borrow from the next, higher column (the place on the left) to complete a subtraction. If we choose to borrow from the next column, we also need to remember to subtract that away when we move on to the next column. Essentially, the current input borrow (what this current column has loaned) is always set to the previous output borrow (what that previous column borrowed).

We should be familiar with decimal subtraction. Let's walk through a simple example of 10 - 5. Starting from the ones' column, we see that we need help to carry out 0 5. Therefore, we **borrow** from the tens' column, and because this is base ten, that means we borrow a value of 10. Now we can successfully carry out 0 - 5 + 10 to get the answer 5. Moving on to the tens column, we need to perform 1 - 0. However, we remember that we previously lent a digit to the ones column, so we have to subtract 1 away: 1 - 0 - 1 = 0. Now, we're done with the problem and we have the complete answer 05.

The same principles apply to binary subtraction. Whenever we borrow from the next place, we borrow a value of 2 because we are in base two.

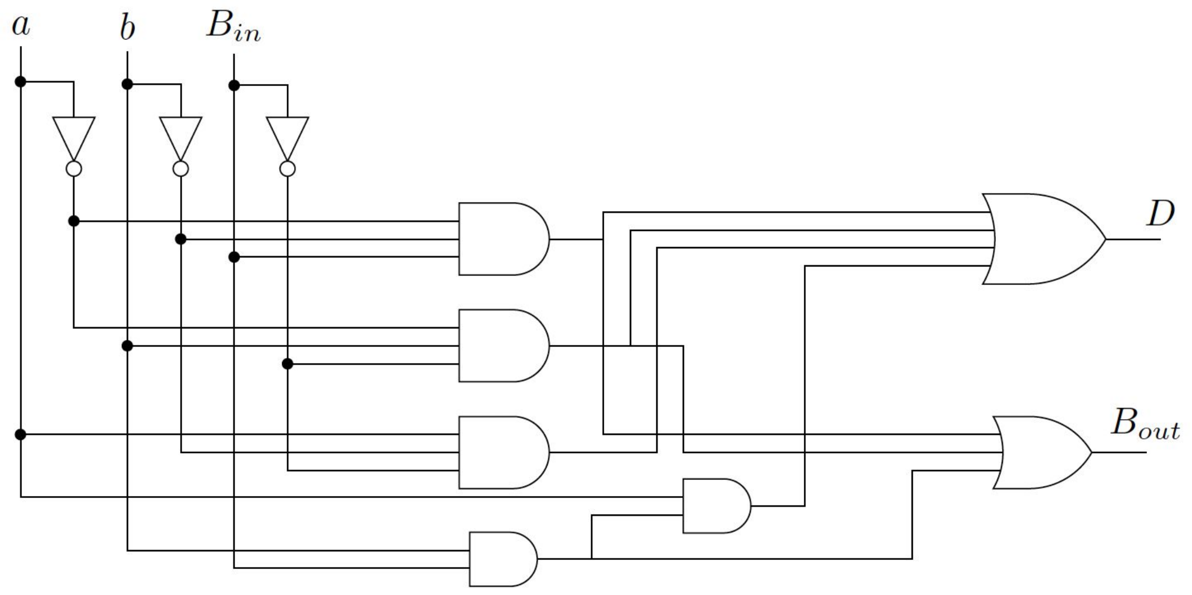
Here is the complete truth table:

a	b	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

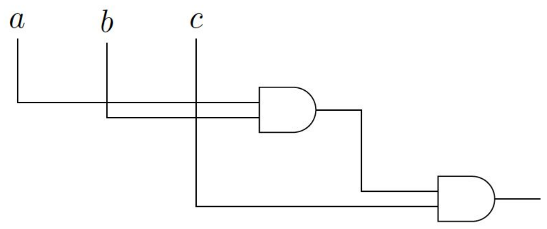
From this truth table, we can derive the boolean expressions for our desired outputs:

$$D = \overline{a}\overline{b}B_{in} + \overline{a}b\overline{B_{in}} + a\overline{b}\overline{B_{in}} + abB_{in}$$
$$B_{out} = \overline{a}\overline{b}B_{in} + \overline{a}b\overline{B_{in}} + \overline{a}bB_{in} + abB_{in} = \overline{a}\overline{b}B_{in} + \overline{a}b\overline{B_{in}} + bB_{in}$$

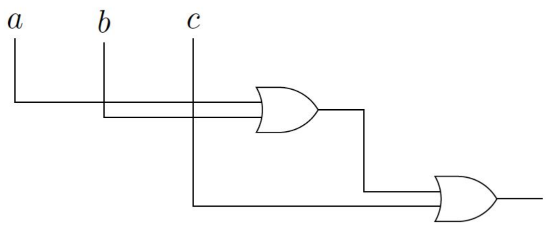
One possible circuit diagram for this circuit is:



An example of a three input AND gate is



An example of a three input OR gate is



An example of a four input OR gate is