1. Two by Two [Lab]

It is HIGHLY recommended/expected that you do this problem BEFORE your lab section!

In this problem we will explore some basic digital circuits, and by the end, you will have designed the circuit you will explore in lab 1. Let's get started!

1. **Binary Addition Warm-Up:** The binary, or base-2, numeral system is a useful abstraction when building digital circuits to support mathematical operations. In the familiar base-10 numeral system, the smallest digit is 0 and the largest digit is 9; however, in binary, there are only 2 digits: 0 and 1. Many of the techniques such as adding and multiplying generalize from base-10 to binary. This problem will introduce you to binary numbers and addition which you will later use to implement a binary adder.

   (a) **Converting binary to base-10.** Each digit of in binary number corresponds to a power of 2. The right-most, or 0th digit corresponds to 2^0; the 1st digit corresponds to 2^1; and the nth digit corresponds to 2^n:

Base-10	Binary
$\underset{10^1}{1}\ \underset{10^0}{3}$	$\underset{2^3}{1}\ \underset{2^2}{1}\ \underset{2^1}{0}\ \underset{2^0}{1}$

   To convert `1101` from binary to base-10, we then multiply each digit by its corresponding power of 2 and sum:

   $$\texttt{1101} = \mathbf{1}\times 2^3 + \mathbf{1}\times 2^2 + \mathbf{0}\times 2^1 + \mathbf{1}\times 2^0 = 13$$

   You try it! Convert `110110` from binary to base-10.

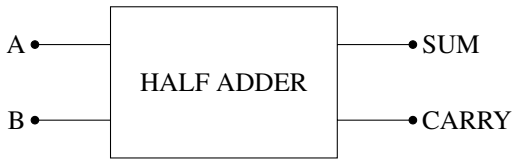
   (b) Now we'll explore how standard addition generalizes from base-10 to binary:

Base-10	Binary
$\begin{array}{r} {\scriptstyle 1\ \ } \\ 19 \\ +15 \\ \hline 34 \end{array}$	$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ 1\ \ } \\ 010011 \\ +001111 \\ \hline 100010 \end{array}$

   You try it! Find the sum of 2 binary numbers `100001` + `110001`. Show your solution in both binary and base-10.

   Note: `11110` is `1111` shifted left logically by one bit. The logical left shift operator, `<<`, therefore is equivalent to multiplication by 2^n. For example, `1 << 1 = 10`, which translates to 2 in decimal, and `1 << 3 = 1000`, which translates to 8 in decimal.

   Introduction to Logical Circuits: the Half-Adder
   The half-adder is a basic digital circuit that adds two one-bit numbers A and B and outputs the sum of those two bits and a carry bit, as shown in the block diagram below:



2. Fill out the following truth table for the half-adder:

A	B	SUM	CARRY
0	0		
0	1		
1	0		
1	1		

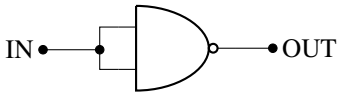
The following are the most common logic gates, aside from the inverter:

							
A	B	AND	OR	NAND	NOR	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

3. Which of the above logic gates is logically equivalent to the SUM column? What about the CARRY column?

4. Design a half-adder by filling in the blanks with the gates you mentioned in the previous part in the logic circuit below.

We can compose logic gates together to form a logic circuit that functions differently from its individual components. Two gates, NAND and NOR, are known as **universal gates**, meaning they can be used to implement any Boolean function without needing to use any other type of gate. Hence, NAND and NOR gates are the basic gates used in all IC digital logic, and are very cheap to fabricate. In fact, AND and OR gates are typically implemented as NAND and NOR gates, respectively, followed by inverters! We will focus on the NAND gate, since it is the canonical universal gate, and the adder circuit you will explore in lab 1 will be made exclusively from NAND gates. In this problem, we will demonstrate that the NAND gate is universal by showing that inverters, AND gates, and OR gates can all be implemented using only NAND gates. For example, this is an inverter made from a NAND gate:



5. Design a logic circuit that performs the same function as an AND gate using only NAND gates.

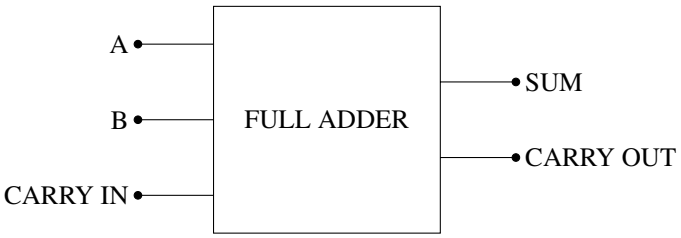
6. Now, to finish the proof, design a logic circuit that performs the same function as an OR gate using only NAND gates.

   Voilà! You have just demonstrated that the NAND gate can be used to implement any Boolean function. Beautiful :')

7. **Just for fun? ;)** The logical equivalent of A XOR B is (A OR B) AND NOT (A AND B). Use this fact to design a logical circuit that functions as an XOR using ONLY NAND gates, inverters, AND gates, and OR gates (or, if you're feeling ∼spicy∼, try to make an XOR using only NAND gates. Hint: the minimal implementation uses 5 NAND gates).

8. Now, to finish up our discussion of the half-adder, provide a brief explanation of how you can build a half adder using only NAND gates.

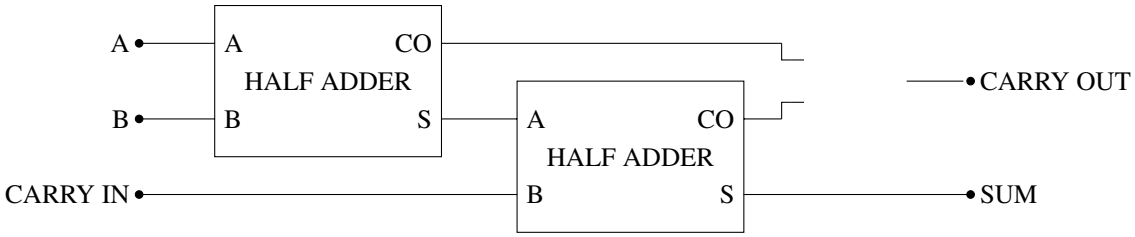
   The full adder is a logic circuit that adds three one-bit numbers A, B, and C, usually split as two one-bit inputs A and B and a carry-in bit C from a previous operation, and outputs two one-bit numbers, a sum and a carry-out bit.



9. Fill out the following truth table for the full adder:

A	B	CARRY IN	SUM	CARRY OUT
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

10. You can make a full adder by logically connecting two half-adders. Fill in the logic gate necessary to make this circuit into a full adder.



   Congratulations! You just designed the logic circuit you will investigate in lab 1.

11. **Bonus:** How could you use that circuit as a building block to make an n-bit adder? How many of that block would you need? How would you connect them?

12. **OPTIONAL:** Try to design a half-adder using only NAND gates. Hint: the minimal implementation uses only 9 gates. The adder circuit you will see in lab 1 will include two of these, connected as in the final non-optional question.