Problem 1 (Formation)

Before you begin the group activity for this week, identify:

- A *Scribe*: who will be responsible for creating a shared document and turning in your solutions for this week's problems.
- A *Timekeeper*: who will keep track of the timer for weekly activities that require timekeeping.
- A *Driver*: who will be responsible for asking people to share or otherwise kickstarting conversation if group chat slows down.

For this problem, simply identify your teammate's roles in the space below.

If you cannot attend a given class period, please work with your group on how you will contribute to the group's work either asynchronously (*e.g.*, by doing the work yourself) or outside of the class period (*e.g.*, by chatting or meeting outside of class). If there are any issues in coordinating work in your group, please let me know as soon as possible!

Problem 2 (There And Back At Again)

In the reading, we introduced a circuit-based model of computation. For these exercises, we'll consider two important ideas related to this model: *universality* and *reversibility*.

1. As a warm-up, spend 5 minutes individually to design a circuit that computes the XOR of two input variables. The XOR function returns 1 if exactly one of its inputs is a 1. (*Hint*: To do this quickly, express XOR in boolean logic in terms of $(\wedge)$, $(\vee)$, and $(\neg)$ and then translate your resulting proposition into a circuit.)
   Come back as a group and compare your answers. Give your final solution in your write-up.
2. Example 9.29 of the book describes how this XOR circuit can be used to implement a family of parity circuits that detect whether an input of size n has an odd number of 1s. With your group, spend 2–3 minutes discussing why this approach implies that the parity function has circuit complexity $\mathcal{O}(n)$.
3. We say that a collection of gates is *universal* if any circuit can be written in terms of these gates. The book presents the circuit complexity model using AND, OR, and NOT gates. These gates are universal for boolean logic, but it turns out we can do even better! Consider the NAND gate. It computes the negation of the AND operator as follows:

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

   Spent 10 minutes individually showing that the NAND gate alone is universal for boolean logic by showing how AND, OR, and NOT can be expressed in terms of NAND. Feel free to assign one gate to each group member. Come back as a group to share and agree upon your solutions, adding them to your write-up.
4. We say that a logic gate f is *reversible* if it is the case that for any output y there is a unique input x such that $f(x) = y$. As a consequence, there must exist an inverse gate f' such that $f(y) = x$. As a group, spend 10–15 minutes answering the following questions about reversible gates:
   - Is OR a reversible gate? Is XOR a reversible gate? Why or why not in each case.
   - Can any of our other binary operation gates be reversible, *e.g.*, AND, OR, NAND, etc.? Why or why not?
   - With an answer to the previous question in mind, design a reversible version of the XOR gate. The gate will have a different number of outputs than the traditional XOR gate but allow you to recover the XOR computation from the result.