This set of exercises is concerned with integers represented in the two's complement notation (see textbook pages 61–64).

Work on the questions below in groups of two or three. After solving each question (or working on it for an agreed amount of time), present and discuss your work with another group. Do you understand the arguments of the other group? Do you find it answers the question?

For concreteness, assume we are working with 8-bit two's complement notation, which can represent integers between -128 and 127. As examples, in this notation 00000101 represents the integer 5 and 10000101 represents the integer $-(2^7) + 5 = -128 + 5 = -123$.

1. Find the representations of the following integers: 0, 7, 27, 127, -128, -121, -101, -1.

2. Argue that in general, a bit sequence $b_7b_6b_5b_4b_3b_2b_1b_0$ represents the integer $b_7 \cdot (-(2^7)) + b_6 \cdot 2^6 + b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$.

3. The algorithm for addition of two numbers in two's complement is described as follows on pages 63–64: Add the two numbers position by position (including any carries), just as for positive integers in standard binary notation. If any carry comes out of the leftmost position (position of b_7), just discard it. If the input numbers are both positive (i.e., both have 0 in their leftmost position), but the result is negative (i.e., has 1 in its leftmost position), or the input numbers are both negative, but the result is positive, report an overflow error. Else return the result.

   Argue that this algorithm is correct, i.e., reports an overflow error when the result cannot be represented in 8-bit two's complement, and else returns the correct result.

   Hint: Start by using the algorithm on various pairs of numbers whose representation you found in question 1. Then argue what happens in general in all eight cases of the possible signs of the two inputs and the single output, while using the expression from question 2.

4. For a 8-bit sequence x, let $\overline{x}$ denote its complement, i.e., the sequence with all bits reversed. As an example, if $x = 10110010$, then $\overline{x} = 01001101$.

   First argue that for any number with two's complement notation x, we have that $x + \overline{x}$ represent the value -1.

Given that addition works correctly (as argued in the preceding question), argue that $x + (\overline{x} + 00000001)$ must represent the value 0.

Argue why this proves that the algorithm on page 62/Figure 1.22 for negating an integer in two's complement notation is correct.

5. Actually, there is exactly one integer where the algorithm does not work. Which is it, and where does the argument in the preceding question break down for this integer?

6. If there is time, work on the following: Recall that any Boolean function can be computed using only AND, OR and NOT gates, since any Boolean function (with one output) can be be written in Disjunctive Normal Form.

   - Show that for any circuit containing AND, OR and NOT gates, we can create an equivalent circuit (producing the same output for any set of inputs) which only has AND and NOT gates. Hint: For each gate in the original circuit, you can replace it by a subcircuit containing only AND and NOT gates. Consider the truth tables.
   - Show that for any circuit containing AND, OR and NOT gates, we can create an equivalent circuit (producing the same output for any set of inputs) which only has NAND gates. Hint: For each gate in the original circuit, you can replace it by a subcircuit containing only NAND gates. Consider the truth tables.