[2] **1** Give an bit blasting transformation for the signed comparisons $\geq_s$ and $>_s$, assuming that negative numbers are represented in two's complement. For example, $\mathbf{7}_4 >_s \mathbf{8}_4$ is supposed to hold, and the constraints $\mathbf{x}_4 \geq_s \mathbf{8}_4$ and $\mathbf{127}_8 \geq_s \mathbf{x}_8$ are valid.

[5] **2** Determine which of the following LLVM compiler optimizations correct, in the sense that the expressions before and after the arrow always correspond to the same value. Try to find a counterexample using an SMT encoding with bit vectors, for bit width 8 and 16.

```
%Op0 = lshr %X, C1
%r = udiv %Op0, C2
=>
%r = udiv %X, C2 << C1
```

```
Pre: isPowerOf2(%Power)
%s = shl %Power, %A
%Y = lshr %s, %B
%r = udiv %X, %Y
=>
%sub = sub %A, %B
%Y = shl %Power, %sub
%r = udiv %X, %Y
```

```
%na = sub 0, %a
%nb = sub 0, %b
%c = add %na, %nb
  =>
%ab = add %a, %b
%c = sub 0, %ab
```

- `Pre` indicates a precondition: the simplification is only applied if the precondition is satisfied. In the encoding, the precondition can therefore be asserted, because one is only interested in counterexamples which satisfy the precondition.
- `lshr` is a logical (unsigned) shift to the right, as provided by `bvlshr` in SMT-LIB.
- `udiv` is unsigned division, as provided by `bvudiv` in SMT-LIB.

[2] ⋆ **3** Give a bit blasting transformation for the left shift $\ll$ and the logical right shift $\gg_u$.

[3] **4** Bit hacks are popular in low-level programming. Prove correctness of the following ones. Let `x` and `y` be bit vectors of size 8.

(a) Show that the line $\mathtt{y} = \mathtt{x}\ \&\!\sim (\mathtt{1} \ll \mathtt{n})$ unsets the nth bit in `x`, for some $0 \leq n \leq 7$: To that end, use an SMT formula with bit vector variables `x` and `y` to express that the nth bit in such an expressions is *not* 0 and show that the formula is unsatisfiable.

(b) The expression $\mathtt{y} = \mathtt{x}\ \&\ (-\mathtt{x})$ is to isolate the right-most 1 bit in `x`, i.e., the right-most 1 bit is kept while all other bits are set to 0. How can we use SMT to check that it does what it should do?

Exercises marked with a ⋆ are optional. Solving them gives bonus points if you submit them before the course via OLAT or email.