Homework

1. (System F) Last week we have seen the encoding of Church numerals, and have *tested* the encoding on example numerals. The goal is now to *prove* the encoding correct. That is, show, in Coq, that your System F representations of addition, multiplication, and exponentiation correspond to their built in versions (on `nat`).

   Remark: for this it may be easier to work with Set instead of Type:

   ```
   Definition cnat := forall X : Set, (X -> X) -> X -> X.
   ```

   - Define a function *nc* from `nat` to `cnat`.
   - Show that this function commutes with each of the operations, e.g. that first adding two `nat`s and then applying *nc*, has the same result as first applying *nc* to both and then adding (in System F) them.
   - Can you define a function *cn* back from `cnat` to `nat`, and show that it really is back, i.e. that the composition of *nc* and *cn* is the identity on `nat`? What about the other composition?

2. Study the development (and if so desired the paper to referred to) in `https://www.irif.fr/~letouzey//download/examples_CiE2008.v` illustrating different ways to extract a division function. Try to adapt *at least one* of the methods to extract a modulo function (i.e. returning the remainder after the division).

   Remark: in my version of Coq, I needed to put parentheses around an occurrence of 'id x' (once) to make the development work (probably some parsing has changed since 2008).