Exercice 1. (Warming up)

1. Inhabit the following types in $\lambda \rightarrow \times$:

\[
X \rightarrow (X \rightarrow R) \rightarrow R \quad A \times ((B \rightarrow R) \rightarrow R) \rightarrow (A \times B \rightarrow R) \rightarrow R
\]

2. Recall the encoding of binary trees with element of type $A$ in system F. How could you generalize it to $n$-ary trees? to infinite branching trees?

Exercice 2 (Typing with type algebra).

In this exercise we give more power to the simple type system by considering types up to some congruence $\equiv$. For example, we can have equivalence of the form $A \equiv A \rightarrow B$.

Typing rules remains unchanged but a type can be replaced by an equivalent type at any point of the derivation. In this system, type derivations are denoted $\Delta \vdash t: A$.

- Show that if $A \equiv A \rightarrow B$ then $\vdash \Omega : B$. (Hint : first show that $\vdash \lambda x.xx : A$)

- Show that if $\equiv$ is a congruence for $\rightarrow$ then the subject reduction property holds. You can use the generation lemma associated to simply typed lambda-calculus.

Exercice 3 (Existential in System F). In propositional second order intuitionistic logic the existential quantifier is introduced and destructed via the following (annotated) rules:

\[
\left(\exists I\right) \quad \Delta \vdash t : T(S/X) \quad \Delta \vdash t : \exists X.T \quad \Delta, x : T \vdash s : B \quad X \not\in FV(\Delta, B) \\
\left(\exists E\right) \quad \Delta \vdash \text{let } [X, x : T] = t \text{ in } s : B
\]

From a logical point of view existentials can be seen as infinite disjunction, from a programming point of view they can be interpreted as an encapsulation mechanism.

1. Find an appropriate type representation for the existential in System F, with an encoding for $\existsI$ and let $\existsE$. Check that it validates the corresponding $\beta$ rule.

2. Recall the encoding of NJ in System F and deduce that second order propositional intuitionistic logic is representable in System F.

3. In programming, streams are co-inductive datatypes with two accessors:

\[
\text{hd} : \text{Str}_A \rightarrow A \quad \text{tl} : \text{Str}_A \rightarrow \text{Str}_A
\]

and a building function $\text{build} : (A \rightarrow B) \rightarrow \text{Str}_A \rightarrow \text{Str}_B$ such that

\[
\text{hd} (\text{build } f \ s) = f(\text{hd}s) \quad \text{tl} (\text{build } f \ s) = \text{build } f (\text{tls})
\]

What could be an encoding of $\text{Str}_A$ in System F?

4. Define the function $\text{nth} : \text{Nat} \rightarrow \text{Str}_A \rightarrow A$ that returns the $n^{th}$ element of a stream.

Exercice 4 (Final 2017 – Equivalence lifting). In HoTT, proofs can become involved. The goal is to prove that if $f : A \rightarrow B$ is an equivalence between $A$ and $B$, then for each pair of elements $a, a' : A$, the map $\text{ap}_f(a, a') : a =_A a' \rightarrow f(a) =_B f(a')$ is an equivalence as well. Let $g : B \rightarrow A$ being an of $f$ meaning that there are witnesses $\alpha : \Pi_{x : A} g(f \ x) =_A \text{id}_A x$ and $\beta : \Pi_{y : B} f(g \ x) =_B \text{id}_B x$ In the sequell we will left the subscript $a, a'$ in $\text{ap}_f$ implicit.
Question 1  As a quasi-inverse candidate for $af$, let us consider $G(\cdot)$, defined by
\[
G(q) \equiv \alpha(a)^{-1} \cdot ap_{q}(q) \cdot \alpha(a')
\] (1)
To satisfy the requirement, we have to exhibit homotopies $\gamma$ (as left inverse) and $\delta$ (as right inverse):
\[
\gamma : \prod_{p : J} G(af(p)) = J p \quad \text{et} \quad \delta : \prod_{q : K} ap_{f}(G(q)) = K q
\]
a) What are the types $J$ and $K$? What is the type of the candidate $G(\cdot)$?
b) Prove the existence of a witness $\gamma$.
c) Why is that not possible to use a similar approach to prove the existence of $\delta$?

Question 2. Let $T : U$ and $\varphi : T \to T$ such that $\varepsilon : \prod_{x : T} \varphi(x) =_{T} \text{id}_{T}(x)$.
a) Given $x, x' : T$ and $r : x =_{T} x'$, prove that $\varepsilon(x)^{-1} \cdot ap_{\varphi}(r) \cdot \varepsilon(x') =_{S} r$, where the type $S$ will be made explicit.
b) Conclude that, for all $x : T$, $\varepsilon(\varphi(x)) = \text{ap}_{\varphi}(\varepsilon(x))$.

Question 3. Given $x : A$, let us note $\nu(x) \equiv \beta(f(x))^{-1} \cdot \beta(f(x))$.
a) State the type of $\nu(\cdot)$.
b) Prove that $\beta(f(a))^{-1} \cdot ap_{f}(ap_{q}(q)) \cdot \beta(f(a')) =_{K} q$.
c) Simplify the path $\nu(a) \cdot ap_{f}(G(q)) \cdot \nu(a')$.
d) Conclude for the existence of an homotopy proof $\delta$ such that
\[
\delta : \prod_{q : K} ap_{f}(G(q)) =_{K} q
\]

Exercice 5 (More on equivalences). Prove the following statements:

1. (identity) For all $A : U$, quasi-inverse ($\text{id}_{A}$) ;
2. (between identity types) For all $A : U$, $x, y : A$ and $p : x =_{A} y$,
   \begin{itemize}
   \item $(p \cdot -) : y = z \to x = z$ and $(p^{-1} \cdot -)$ are quasi-inverse one of the other ;
   \item $(- \cdot p) : z = x \to z = y$ et $(- \cdot p^{-1})$ are quasi-inverse one of the other.
   \end{itemize}
3. (transport) If $P : A \to U$, then $\text{tr}^{P}(p, -) : P(x) \to P(y)$ has $\text{tr}^{P}(p^{-1}, -)$ for a quasi-inverse.

Exercice 6 (Barendregt natural numbers). Back to pure $\lambda$-calculus The Barendregt natural numbers $[n]$ ($n \in \mathbb{N}$) are defined by:
\[
[0] \equiv \textbf{I} \quad [n + 1] \equiv \textbf{(pair F} [n])
\]
a) Using the alternative representation, code the successor, predecessor and test-to-zero functions.
b) Implement the addition.
c) In your understanding, how to the two natural numbers encodings (Church vs Barendregt) compare?