1. **Some more tautologies.** (10 points)
   Let $A : \mathcal{U}$ and $P, Q : A \to \mathcal{U}$ be dependent types. Show that
   (a)
   $$\left(\prod_{x:A} P(x) \times Q(x)\right) \longrightarrow \left(\prod_{x:A} P(x)\right) \times \left(\prod_{x:A} Q(x)\right),$$
   (b)
   $$\left(\sum_{x:A} P(x)\right) + \left(\sum_{x:A} Q(x)\right) \longrightarrow \left(\sum_{x:A} P(x) + Q(x)\right).$$

2. **The operad of binary trees.** (10 points)
   (a) Construct a function $\texttt{leaves} : \texttt{BinTree} \to \mathcal{U}$ which maps every binary tree to its collection of leaves. (A *leaf* in a binary tree is a node without any outgoing edges.)
   (b) Let $m : A \to A \to A$ be a magma structure[1] on a type A. Whenever we label the leaves of a binary tree by elements of A, we can consider this labeled tree as a prescription for successively multiplying the leaf labels together via m, with the result being another element of A. Construct a function which turns every labeled binary tree into an element of A like this.

   Hint: If you do this in Coq, use this to define the type of binary trees:

```
Inductive BinTree : Type :=
| treeroot : BinTree
| join : BinTree → BinTree → BinTree.
```

3. **The type-theoretic principle of choice.** (10 points)
   For $A, B : \mathcal{U}$, a *binary relation* is a function $R : A \to B \to \mathcal{U}$. Show that
   $$\left(\prod_{x:A} \sum_{y:B} R(x,y)\right) \longrightarrow \left(\sum_{f:A\to B} \prod_{x:A} R(x, f(x))\right).$$

   Note: This does not imply the set-theoretic axiom of choice, which we will discuss later on.

4. **Induction, induction.** (10 points)
   (a) Construct the predecessor function $\texttt{pred} : \mathbb{N} \to \mathbb{N}$. (You can take the predecessor of 0 to be 0.)
   (b) For $A, B : \mathcal{U}$, construct a function
   $$(A \to B) \to (\texttt{List}(A) \to \texttt{List}(B))$$
   which takes a function and applies it to each element in a list. (This is Haskell's `map`.)
   (c) For $A : \mathcal{U}$, define a type $\texttt{BinTree}(A)$ of binary trees with nodes labeled by elements of A. Construct a function $\texttt{BinTree}(A) \to \texttt{List}(A)$ which lists all the labels of such a tree in the order of your choice.

[1] Recall that a magma structure is simply a binary operation not required to satisfy any particular law.