1. Write two functions

\[ \text{beep} : (\text{Pair } a \ b \rightarrow c) \rightarrow (a \rightarrow b \rightarrow c) \]

\[ \text{boop} : (a \rightarrow b \rightarrow c) \rightarrow (\text{Pair } a \ b \rightarrow c) \]

such that for every function \( f : \text{Pair } a \ b \rightarrow c \) the equation

\[ \text{boop} (\text{beep } f) = f \]

holds, and for every function \( g : a \rightarrow b \rightarrow c \) the equation

\[ \text{beep} (\text{boop } g) = g \]

holds.

2. (a) Write functions

\[ \text{conjunction} : \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool} \]

\[ \text{disjunction} : \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool} \]

that compute the logical conjunction and disjunction, respectively, of their inputs.

(b) Using \text{foldList}, write a function \( \text{conj} : \text{List } \text{Bool} \rightarrow \text{Bool} \) that returns the logical conjunction of the entire input list.

(c) Using \text{foldList}, write a function \( \text{disj} : \text{List } \text{Bool} \rightarrow \text{Bool} \) that returns the logical disjunction of the entire input list.

(d) Write the filter function for lists, \( \text{filterList} : (a \rightarrow \text{Bool}) \rightarrow \text{List } a \rightarrow \text{List } a \) in terms of \text{foldList}.

3. Recall the type of binary trees:

\[
\begin{align*}
\text{data Tree} & : \text{Type} \rightarrow \text{Type} \text{ where} \\
\text{Leaf} & : \text{Tree } a \\
\text{Node} & : \text{Tree } a \rightarrow a \rightarrow \text{Tree } a \\
\end{align*}
\]
(a) Write the fold function for binary trees.

(b) Use the fold function for binary trees to write the map function for binary trees, \( \text{mapTree} : (a \to b) \to \text{Tree } a \to \text{Tree } b \).

(c) Use the fold function for binary trees to write a function \( \text{sumTree} : \text{Tree } \text{Nat} \to \text{Nat} \) that sums the data in the input tree.