14 Types (NK)

(a) Give typing rules for the introduction form \( \text{pack}(\tau, M) \) and
elimination form \( \text{unpack} M \text{ as } (x, \alpha) \text{ in } N \) of the existential type \( \exists \alpha(\tau) \).

[4 marks]

(b) An infinite stream of booleans can be represented in the polymorphic lambda
    calculus using the existential type

\[
\text{stream} \triangleq \exists \alpha (\alpha \times (\alpha \rightarrow (\text{bool} \times \alpha)))
\]

(i) Using the encoding above, define a function \( \text{head} : \text{stream} \rightarrow \text{bool} \).

[3 marks]

(ii) Using the encoding above, define a function \( \text{tail} : \text{stream} \rightarrow \text{stream} \).

[3 marks]

(iii) Using the encoding above, define a function

\[
\text{unfold} : \forall \alpha (\alpha \rightarrow (\alpha \rightarrow (\text{bool} \times \alpha)) \rightarrow \text{stream})
\]

[4 marks]

(iv) Using \( \text{unfold} \) and the other functions above, define a function \( \text{notstream} : \text{stream} \rightarrow \text{stream} \), which returns a stream containing the boolean negation
    of the elements of the input stream. This answer should not use explicit
    \( \text{pack} \) or \( \text{unpack} \) expressions.

[6 marks]

[Note: You may use extensions to the pure polymorphic lambda calculus such
    as let-bindings, natural numbers, products, and sum types, but carefully note
    their use and their typing in your answers.]