1 Let-polymorphism [80 points]

The following shows the abstract syntax for the let-polymorphism system:

- **Monotype**
  \[ A ::= A \rightarrow A \mid \alpha \]

- **Polytype**
  \[ U ::= A \mid \forall \alpha. U \]

- **Expression**
  \[ e ::= x \mid \lambda x : A. e \mid e e \mid \Lambda \alpha. e \mid e \sqbracket{A} \mid \text{let } x : U = e \text{ in } e \]

**Question 1. [30 points]** An erasure function \( \text{erase}(\cdot) \) takes an expression in the let-polymorphism system and erases all type annotations in it to produce a corresponding expression in the implicit let-polymorphism system, i.e., the let-polymorphism without type annotations. Complete the definition of the erasure function \( \text{erase}(\cdot) \).

\[
\begin{align*}
\text{erase}(x) &= x \\
\text{erase}(\lambda x : A. e) &= \lambda x. \text{erase}(e) \\
\text{erase}(e_1 e_2) &= \text{erase}(e_1) \text{ erase}(e_2) \\
\text{erase}(\Lambda \alpha. e) &= \text{erase}(e) \\
\text{erase}(e \sqbracket{A}) &= \text{erase}(e) \\
\text{erase}(\text{let } x : U = e \text{ in } e') &= \text{let } x = \text{erase}(e) \text{ in } \text{erase}(e')
\end{align*}
\]

**Question 2. [20 points]** Give a well-typed closed expression \( e \) in the let-polymorphism system such that \( \text{erase}(e) = \text{let } f = \lambda x. x \text{ in } (f \text{ true, } f \text{ 0)} \). Assume two monotypes bool for boolean values and int for integers.

\[
\text{let } f : \forall \alpha. \alpha \rightarrow \alpha = \Lambda \alpha. \lambda x : \alpha. x \text{ in } (f \sqbracket{\text{bool}} \text{ true, } f \sqbracket{\text{int}} 0))
\]

**Question 3. [30 points]** The let-polymorphism system uses a typing judgment \( \Gamma \vdash e : U \) where \( \Gamma \) is a typing context defined as follows:

\[
\text{typing context} \quad \Gamma ::= \cdot \mid \Gamma, x : U \mid \Gamma, \alpha \text{ type}
\]

The implicit let-polymorphism system, i.e., the let-polymorphism without type annotations, uses a new typing judgment \( \Gamma \triangleright e : U \) which states that untyped expression \( e \) is typable with a polytype \( U \).

State the soundness and completeness (i.e., the correctness) of the new typing judgment \( \Gamma \triangleright e : U \) with respect to the typing judgment \( \Gamma \vdash e : U \).

**Theorem 1.1** (Soundness). If \( \Gamma \triangleright e : U \), then there exists a typed expression \( e' \) such that \( \Gamma \vdash e' : U \) and \( \text{erase}(e') = e \).

**Theorem 1.2** (Completeness). If \( \Gamma \vdash e : U \), then \( \Gamma \triangleright \text{erase}(e) : U \).

2 Type reconstruction

**Question 4. [20 points]** Assume the type system with implicit let-polymorphism (which we use for the type reconstruction algorithm). What is the most general type that you can assign to \( \lambda x. x \)?