1 System F

Question 1. [10 points] The definition of ordered typing contexts in System F as follows:

typing context \( \Gamma ::= \cdot | \Gamma, x : A | \Gamma, \alpha \) type

Explain why we need ordered typing contexts (instead of unordered typing contexts) in the definition of System F.

Answer: A type \( A \) involving type variable \( \alpha \) may be invalid if \( \alpha \) has not been declared before. In order to keep track of which type variables have been declared before, we use ordered typing contexts.

Question 2. [30 points] Complete the definitions of monotypes, polytypes, expressions, and values in the abstract syntax of the predicative polymorphic \( \lambda \)-calculus:

- **monotype** \( A ::= A \rightarrow A | \alpha \)
- **polytype** \( U ::= A | \forall \alpha. U \)
- **expression** \( e ::= x | \lambda x : A. e | e e | \Lambda \alpha. e | e [A] \)
- **value** \( v ::= \lambda x : A. e | \Lambda \alpha. e \)

Question 3. [20 points] Explain why type reconstruction in System F is undecidable. Also explain why the predicative polymorphic \( \lambda \)-calculus recovers the decidability of type reconstruction.

Answer: In System F, type variable \( \alpha \) ranges over all types, including polymorphic types \( \forall \alpha . A \) themselves. Since we introduce type variables in order to use polymorphic types \( \forall \alpha . A \), there exists a certain form of circularity in the definition of types, which is the source of the undecidability of type reconstruction in System F. In the case of the predicative polymorphic \( \lambda \)-calculus, such a circularity disappears and type reconstruction becomes decidable.

Question 4. [20 points] Explain the notions of predicativity and impredicativity.

Answer: Suppose that we have a set \( S \) of elements. We would like to introduce quantifications over the set \( S \). Let us introduce a new form of quantification \( \forall \alpha. \) over the set \( S \) where \( \alpha \) is a variable that ranges over \( S \). If the domain of \( \alpha \) includes \( \forall \alpha. \), the resultant system is called impredicative. If not, i.e., the domain of \( \alpha \) includes only the original set \( S \), the resultant set is called predicative.

Question 5. [20 points] Suppose that we have a subtyping system for a fictional programming language \( \Xi \). The subtyping system uses a subtyping judgment \( A \leq B \) where \( A \) and \( B \) are types in \( \Xi \). Now we have implemented the subtyping system in Standard ML. Our implementation provides a function \( \text{subtype} : \text{typ} \rightarrow \text{typ} \rightarrow \text{bool} \) where \( \text{typ} \) is the type of values representing types in \( \Xi \). Informally \( \text{subtype} A B \) returns \( \text{true} \) if and only if \( A \leq B \) holds.

Explain what it means that \( \text{subtype} \) is sound.

Explain what it means that \( \text{subtype} \) is complete.

Answer: \( \text{subtype} \) is sound if \( \text{subtype} A B \) returns \( \text{true} \) implies \( A \leq B \); that is \( \text{subtype} \) always returns correct results. \( \text{subtype} \) is sound if \( A \leq B \) implies that \( \text{subtype} A B \) returns \( \text{true} \); that is \( \text{subtype} \) misses no subtyping relations.