1. (16.2.5) By induction on declarative typing derivations. Proceed by cases on the final rule in the derivation.

   - *Case TVar*: $t = x$ and $\Gamma(x) = T$. Immediate, by $TAVar$.
   - *Case TAbs*: $t = \lambda x : T_1.t_2$, $\Gamma, x : T_1 \vdash \ t_2 : T_2$ and $T = T_1 \to T_2$. By the induction hypothesis, $\Gamma, x : T_1 \Vdash \ t_2 : S_2$, for some $S_2 <: T_2$. By $SArrow$, $T_1 \to S_2 <: T_1 \to T_2$.
   - *Case TApp*: $t = t_1 t_2$, $\Gamma \vdash t_1 : T_{11} \to T_{12}$, $\Gamma \vdash t_2 : T_{11}$ and $T = T_{11}$. By the induction hypothesis, $\Gamma \Vdash t_1 : S_1$, for some $S_1 <: T_{11} \to T_{12}$ and $envEt_2 : S_2$, for some $S_2 <: T_{11}$. By the inversion lemma for the subtype relation, S_1 must have the form $S_{11} \to S_{12}$, for some S_{11} and S_{12} with $T_{11} <: S_{11}$ and $S_{12} <: T_{12}$. By transitivity, $S_2 <: S_{11}$. By the completeness of algorithmic subtyping, $\Vdash S_2 <: S_{11}$. Now, by $TAApp$, $\Gamma \Vdash t_1 t_2 : S_{12}$, which finishes this case (since we already have $S_{12} <: T_{12}$).
   - *CaseTRcd*: $t = \{l_i = {t_i}^{i \in 1 \cdots n}\}$, $\Gamma \vdash t_i : T_i$ for each i and $T = \{l_i : {T_i}^{i \in 1 \cdots n}\}$. By the induction hypothesis, $\Gamma \Vdash t_i : S_i$, with $S_i <: T_i$ for each i. Now, by $TARcd$, $\Gamma \Vdash \{l_i = {t_i}^{i \in 1 \cdots n}\} : \{l_i : {S_i}^{i \in 1 \cdots n}\}$. By $SRcd$ $\{l_i : {S_i}^{i \in 1 \cdots n}\} <: \{l_i : {T_i}^{i \in 1 \cdots n}\}$, which finishes this case.
   - *TProj*: $t = t_1.l_j$, $\Gamma \vdash t_1 : \{l_i : T_i^{i \in 1 \cdots n}\}$ and $T = T_j$. By the induction hypothesis, $\Gamma \Vdash t_1 : S$, with $S <: \{l_i : T_i^{i \in 1 \cdots n}\}$. By the inversion lemma in subtyping relation, S must have the form $\{k_i : S_i^{i \in 1 \cdots m}\}$, with at least the labels $\{{l_i}^{i \in 1 \cdots n}\}$ i.e., $\{l_i^{i \in 1 \cdots n}\} \subseteq \{{k_j}^{j \in 1 \cdots m}\}$, with $S_j <: T_i$ for each common label $l_i = k_j$. By $SARcd$ $\Gamma \Vdash t_1.t_j : S_j$, which finishes this case (since we already have $S_j <: T_j$).
   - *Case TSub*: $\Gamma \vdash t : S$ and $S <: T$. By the induction hypothesis and transitivity of subtyping.

2. (16.2.6) If we dropped the arrow subtyping rule $S - Arrow$ but kept the rest of the declarative subtyping and typing rules the same, the system do not still have the minimal typing property. For example, the term $\lambda x : \{a : Nat\}.x$ has both the types $\{a : Nat\} \to \{a : Nat\}$ and $\{a : Nat\} \to Top$ under the declarative rules. With the algorithmic typing has the type $\{a : Nat\} \to \{a : Nat\}$, but without $S - Arrow$, this type is incomparable with $\{a : Nat\} \to Top$.

3. (16.3.3) The minimal type of if *true* then *false* else { } is *Top*, the join of *Bool* and { }. However, it is hard to imagine that the programmer really intended to write this expression, after all, no operations can be performed on a value of type Top.

4. (14.3.1)

Syntax			
t	::=		*terms* :
		x	*variable*
		$\lambda x : T.t$	*abstraction*
		$t\ t$	*application*
		rise $(< l = t >$ as $T)$	*rise exception*
		try t with $\lambda x : T.$ case x of $< l = x > \ \Rightarrow h$ $\mid \_ \Rightarrow$ rise x	*handle exception*

$$\frac{\Gamma \vdash t_j : T_j \quad T_{exn} = \{l_i : T_i^{i \in 1 \cdots n}\}}{\mathsf{rise}\ (< l_j = t_j > \mathsf{as}\ T_{exn})} (\text{TExn})$$

$$\frac{\Gamma \vdash t : T \quad T_{exn} = \{l_i : T_i^{i \in 1 \cdots n}\} \quad \Gamma, x_j : T_j \vdash\ h : T}{\begin{array}{l}\Gamma \vdash \mathsf{try}\ t\ \mathsf{with}\ \lambda e : T_{exn}.\ \mathsf{case}\ e\ \mathsf{of} \\ < l_j = x_j > \ \Rightarrow h \\ |\ \_ \Rightarrow \mathsf{rise}\ e : T\end{array}} (\text{TTry})$$

$(\mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn}))\ t_2 \longrightarrow \mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})$ (EAppRaise1)

$v\ (\mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})) \longrightarrow \mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})$ (EAppRaise2)

$$\frac{t_1 \longrightarrow t_1'}{\mathsf{rise}\ t_1 \longrightarrow \mathsf{rise}\ t_1'} (\text{ERaise})$$

$\mathsf{rise}\ (\mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})) \longrightarrow \mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})$ (ERaiseRaise)

$$\frac{t_j \longrightarrow t_j'}{\mathsf{rise}\ (< l_j = t_j > \ \mathsf{as}\ T_{exn}) \longrightarrow \mathsf{rise}\ (< l_j = t_j' > \ \mathsf{as}\ T_{exn})} (\text{ERaiseVariant})$$

$\mathsf{try}\ v\ \mathsf{with}\ t_2 \longrightarrow v$ (ETryV)

$\mathsf{try}\ \mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})\ \mathsf{with}\ \lambda e : T_{exn}.\ \mathsf{case}\ e\ \mathsf{of}$
$< l_j = x_j > \ \Rightarrow h$
$|\ \_ \Rightarrow \mathsf{rise}\ e \longrightarrow [x_j \mapsto v_j]h$ (ETryRise1)

$$\frac{l_j \neq l_k}{\begin{array}{l}\mathsf{try}\ \mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})\ \mathsf{with}\ \lambda e : T_{exn}.\ \mathsf{case}\ e\ \mathsf{of} \\ < l_j = x_j > \ \Rightarrow h \\ |\ \_ \Rightarrow \mathsf{rise}\ e \longrightarrow \mathsf{rise}\ (< l_j = v_j > \ \mathsf{as}\ T_{exn})\end{array}} (\text{ETryRise2})$$

$$\frac{t_1 \longrightarrow t_1'}{\mathsf{try}\ t_1\ \mathsf{with}\ t_2 \longrightarrow \mathsf{try}\ t_1'\ \mathsf{with}\ t_2} (\text{ETry})$$