Sample solution for Pierce 15.5.2.

(1) If we drop the first premise of S-REF, we have

$$\frac{\mathtt{T}_1 <: \mathtt{S}_1}{\mathtt{Ref}\ \mathtt{S}_1 <: \mathtt{Ref}\ \mathtt{T}_1} \qquad \text{(S-REF1)}$$

Since $\{\mathtt{x:Bool,y:Bool}\} <: \{\mathtt{x:Bool}\}$, we have by S-REF1 that $\mathtt{Ref}\ \{\mathtt{x:Bool}\} <: \mathtt{Ref}\ \{\mathtt{x:Bool,y:Bool}\}$. So the program

```
(λr:Ref {x:Bool,y:Bool}.!r.y)(ref {x=true})
```

or, equivalently,

```
let r = ref {x=true} as Ref {x:Bool,y:Bool} in
   !r.y
```

will typecheck, since we can apply T-SUB to the argument to make it match the declared type (or ascription) for `r`. But the program will clearly get stuck attempting to access the `y` field from a record that has none.

(2) If we drop the second premise of S-REF, we have

$$\frac{\mathtt{S}_1 <: \mathtt{T}_1}{\mathtt{Ref}\ \mathtt{S}_1 <: \mathtt{Ref}\ \mathtt{T}_1} \qquad \text{(S-REF2)}$$

This gives $\mathtt{Ref}\ \{\mathtt{x:Bool,y:Bool}\} <: \mathtt{Ref}\ \{\mathtt{x:Bool}\}$. So the program

```
let r = ref {x=true,y=true}
in (λu:Ref {x:Bool}. u := {x=true}) r;
   r.y
```

or, equivalently,

```
let r = ref {x=true,y=true} in
let u = r as Ref {x:Bool} in
    u := {x=true};
    r.y
```

will typecheck, since we can apply T-SUB to `r` to make it match the declared type (or ascription) for `u`. But the program will again get stuck attempting to access the `y` field from a record that has none.

Sample solution for Pierce 16.2.3.

Let $\mathtt{s} = (\lambda \mathtt{r:S.r})\{\mathtt{x=true,y=true}\}$, $\mathtt{t} = \{\mathtt{x=true,y=true}\}$, $\mathtt{S} = \{\mathtt{x:Bool}\}$, and $\mathtt{T} = \{\mathtt{x:Bool,y:Bool}\}$. Then we have $\mathtt{s} \to^* \mathtt{t}$, $\vdash \mathtt{s} : \mathtt{S}$, $\vdash \mathtt{t} : \mathtt{T}$, and $\mathtt{T} <: \mathtt{S}$ but not vice-versa.