Sample solution for Pierce 15.5.2.

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

\[
\frac{T_1 <: S_1}{\text{Ref } S_1 <: \text{Ref } T_1} \quad \text{(S-REF1)}
\]

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

\[
(\lambda r: \text{Ref } \{x:\text{Bool}, y:\text{Bool}\}. \!r.y)(\text{ref } \{x=\text{true}\})
\]

or, equivalently,

\[
\text{let } r = \text{ref } \{x=\text{true}\} \text{ as Ref } \{x:\text{Bool}, y:\text{Bool}\} \text{ in } \\
\text{!r.y}
\]

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

\[
\frac{S_1 <: T_1}{\text{Ref } S_1 <: \text{Ref } T_1} \quad \text{(S-REF2)}
\]

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

\[
\text{let } r = \text{ref } \{x=\text{true}, y=\text{true}\} \\
\text{in } (\lambda u: \text{Ref } \{x:\text{Bool}\}. \ u := \{x=\text{true}\}) \ r; \\
\ r.y
\]

or, equivalently,

\[
\text{let } r = \text{ref } \{x=\text{true}, y=\text{true}\} \text{ in } \\
\text{let } u = r \text{ as Ref } \{x:\text{Bool}\} \text{ in } \\
\ \ u := \{x=\text{true}\}; \\
\ r.y
\]

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 15.5.3

Compiling and executing the program

class T {
    public static void main(String argv[]) {
        Object[] a = new String[1];
        a[0] = new Object();
    }
}

yields

Exception in thread "main" java.lang.ArrayStoreException
    at T.main(T.java:4)