This assignment requires no programming, just written responses. To help the graders, your answers should be typed up rather than handwritten.

1. [20%] For the first 4 parts, give the most general conditions on $\text{type}_1$ and $\text{type}_2$ for the given subtyping judgments to be “safe” (not go wrong at run-time) in an ML-like language with subtyping: $\text{type}_1 \preceq \text{type}_2$ or $\text{type}_1 \succeq \text{type}_2$ or $\text{type}_1 = \text{type}_2$. Give a brief and clear explanation.

   (a) $\text{type}_1 \text{ list} \preceq \text{type}_2 \text{ list}$.
   (b) $\text{type}_1 \text{ foo} \preceq \text{type}_2 \text{ foo}$, defined by $\text{type } 'a \text{ foo } = 'a \text{ list } * 'a \text{ list}$.
   (c) $\text{type}_1 \text{ bar} \preceq \text{type}_2 \text{ bar}$, defined by $\text{type } 'a \text{ bar } = 'a \rightarrow 'a$
   (d) $\text{type}_1 \text{ baz} \preceq \text{type}_2 \text{ baz}$, defined by $\text{type } 'a \text{ baz } = ('a \rightarrow \text{int} ) \rightarrow \text{int}$.

2. [20%] If $\text{type}_1 \preceq \text{type}_2$ then we know from class that neither $\text{type}_1 \text{ ref}$ nor $\text{type}_2 \text{ ref}$ is a safe subtype of the other (at least without run-time checks). However, is it safe to assign a value of type $\text{type}_1$ to a $\text{type}_2 \text{ ref}$ or a value of type $t_2$ to a $t_1 \text{ ref}$? Again, give a brief explanation of your answer.

3. In all of the following cases, assume that we have a class Food and two subclasses (and subtypes) of Food, Vegetable and Meat.

   (a) [20%] The C++ language allows a subclass to override a method inherited from the superclass with code having a more specific return type. Thus, we could have a class Animal having a method

   \[
   \text{virtual Food favoriteFood()}
   \]

   and a subclass Cow which replaces this with a method having type

   \[
   \text{virtual Vegetable favoriteFood()}
   \]

   guaranteeing that the favorite food of any cow will be not just some food, but a vegetable.
Is it always safe to make \texttt{Cow} be a subtype of \texttt{Animal}? That is, it it still safe to provide a (pointer to a) \texttt{Cow} object where a (pointer to an) \texttt{Animal} object is expected? Here “safe” means that the typechecker rejects code that would try to invoke methods appearing only in a subclass using an object which was created by a superclass (and hence doesn’t have such methods).

(b) [20\%] The Eiffel language, developed by Bertrand Meyer, is another object-oriented language containing many features aimed at large-scale software development. (For example, one can specify preconditions, postconditions, and invariants assertions that get checked at run-time.)

The Eiffel language allows a subclass to \textit{override} a method inherited from the superclass with code having a \textit{more specific} argument types.

Thus we can specify a class \texttt{Animal} with a method (still using C++ syntax, since you probably don’t know Eiffel)

\begin{verbatim}
    virtual void eat(Food* f)
\end{verbatim}

and then \textit{override} this in a subclass \texttt{Cow} with a method having type

\begin{verbatim}
    virtual void eat(Vegetable* v)
\end{verbatim}

specifying that it is only correct for cows to eat vegetables, rather than arbitrary code.

Show that if subclasses are subtypes, then one can write Eiffel code that type-checks but at run-time feeds meat to a cow (potentially causing the program to crash).

(c) [20\%] If a language used the opposite rule from Eiffel, so that subclasses could replace methods with code allowing less-specific arguments, would this be safe? For example, assume the class \texttt{Animal} has a method

\begin{verbatim}
    virtual void eat(Vegetable* f)
\end{verbatim}

and then \textit{override} this in a subclass \texttt{Omnivore} with a method having type

\begin{verbatim}
    virtual void eat(Food* v)
\end{verbatim}