1. An `int` in Java is represented as a 32-bit number, `float` and `double` are represented as 32-bit and 64-bit IEEE 754 floating point numbers, respectively. Suppose we define the subtype relationship on integers and floating point numbers as a subset relationship on the sets of numbers that can be represented in a given type. Is `int` a subtype of `float`? Is it a subtype of `double`?

You can easily find the precise layout of IEEE floating points by searching for ‘IEEE 754’ on Google or other search engines.

2. Given the following Java code:
   ```java
   class C {
     public int foo (C x) { return 0; }
   }
   class D extends C {
     public int foo (C x) { return 1; }
     public int foo (D x) { return 2; }
   }

   C p = new D();
   C q = new D();
   int i = p.foo(q);
   ```

Which method is executed for the call `p.foo(q)`? Explain why.

3. Explain in English what the ML type
   ```ml
   ('a -> 'b -> 'c) -> ('d -> 'e) -> ('a * 'd) list -> (('b -> 'c) * 'e) list
   ```
   stands for.

4. Suppose we allow subtyping for function types. E.g., a variable binding of the form (in ML syntax)
   ```ml
   val f : s -> t = g;
   ```
   would be legal if the type of function `g` is a subtype of `s->t`. I.e., `g` must be able to handle any argument of type `s` and produce a result that can be assigned to a variable of type `t`.

Assume `t` is a subtype of `s`. Which of the following function types is a subtype of which other function type?
5. Translate the following ML code into a C++ or Java class hierarchy.

(* A tree is either a Leaf containing an integer value or an interior Node with two subtrees. *)

datatype Tree = Leaf of int
               | Node of Tree * Tree

(* Return the sum of all the integers stored in Leaf nodes. *)

fun sum (Leaf i)   = i
    | sum (Node (l, r)) = sum l + sum r

Define the class hierarchy with classes Tree, Leaf, and Node, such that the following code works (in C++ syntax):

    Tree * left = new Leaf(1);
    Tree * right = new Node(new Leaf(2), new Leaf(3));
    Tree * root = new Node(left, right);
    int      h = root->sum();

where sum() is a virtual function. Do not use an if-then-else.