CS 386L (COOK)  Take home Due May 13 at noon

You may (1) use your notes, (2) the lecture notes delivered in class, (3) Pierce (including any solutions in the back of the book) and (4) http://zvon.org/comp/r/ref-Haskell.html (only files on zvon) (5) papers linked from the class schedule, but nothing else. A clear, short answer is better than a long, complex, or vague one.

1. In a type system with records, unions, functions, and universal/existential types...
   a. Is there a set of types that are an infinite descending chain of subtypes? That is a set of distinct types \( \ldots \triangleleft: t_3 \triangleleft: t_2 \triangleleft: t_1 \triangleleft: t_0 \)? If so, describe such a set.
   b. Is there an infinite ascending chain of types? If so, describe it.
   c. Is there a chain that is both ascending and descending? If so, describe it.

2. a) Write a short program that will fail with a runtime type error (i.e. its evaluation will get stuck) if the first premise of S-Ref is dropped (Exercise 15.5.2(1))

b) Write a program that will fail if the second premise is dropped (Exercise 15.5.2(2))

3. Give your answer in System F from the book. Let \( \textsf{Pair} \) be a type function that defines the type of Church pairs:
   \[ \textsf{Pair}[A, B] = \forall R. (A \to B \to R) \to R \]
   Let \( \textsf{pair} \) create a Church pair:
   \[ \textsf{pair} : \forall A. \forall B. A \to B \to \textsf{Pair}[A, B] \]
   Write a function \( \textsf{swap} \) that takes types \( A \) and \( B \), and a \( \textsf{Pair}[A, B] \) as input and produces a \( \textsf{Pair}[B, A] \) with the two elements swapped. Be careful to give all explicit type arguments.
   (For example to call the identity function \( \text{id} : \forall A. A \to A \) on an integer you must use the form \( \text{id}[\text{Int}] 3 \). Give the type and function definition of \( \text{swap} \).

4. Assume that \( P \) is an ADT implementation with the following signature:
   \[ P : \exists t. \{ z: t, s: t \to t, m: t \to t \to t, d: t \to \text{Nat} \} \]
   \[ P = \{ ^*\text{Nat}, \{ z=0, s=\text{succ}, m=\text{max}, d=\text{id[Nat]} \} \} \]
   where \( 0, \text{succ}, \text{max} \) are the standard operations on natural numbers.
   Explain whether or not the following programs are legal:
   a) \( \text{let} \{ X, p \} = P \text{ in } p.s(p.z) \)
   b) \( \text{let} \{ X, p \} = P \text{ in } p.m(p.s(p.z))(\text{succ}(0)) \)
   c) \( \text{let} \{ X, p \} = P \text{ in } p.m(p.d(p.z))(p.z) \)
   d) \( \text{let} \{ X, p \} = P \text{ in } \text{let} \{ Y, r \} = P \text{ in } p.m(r.z)(p.z) \)
   e) \( \text{let} \{ X, p \} = P \text{ in } p.d(p.s(p.z)) \)

5. (a) In your own words, what is a monad?
   (b) How is a monad used?
6. Give your answer in Haskell. Assume that $M \ t$ is a monad.
Define the following functions using standard monadic return and $>>=\$

a) A function to compose two computations:
   \[ \text{comp :: } (a \to M \ b) \to (b \to M \ c) \to (a \to M \ c) \]

b) A function that takes a computation that returns a computation; it first runs the outer computation then returns the result, which is another computation:
   \[ \text{flop :: } M \ (M \ a) \to M \ a \]

c) Define a function that takes an ordinary function of type $a \to b$, and moves it into the monad. The result is a function that takes a computation producing $a$s and returns a computation producing $b$s.
   \[ \text{flip :: } (a \to b) \to (M \ a \to M \ b) \]

7. Specify all subtyping relationships between the following types:
   \[
   \{ \text{val: Nat} \} \\
   < \text{val: Nat} > \\
   \text{Top} \\
   < \text{val: Nat, fun: Bool} > \\
   \mu X. < \text{val: Nat, op: } X \to X, \text{fun: Bool} > \\
   \mu X. \{ \text{val: Nat, op: } X \to X \} \\
   \mu X. \{ \text{val: Nat, op: } \text{Top} \to X \}
   \]

8. Consider the following program, which computes the result of a polynomial given a list of coefficients.

   \[
   \text{pow n x = if n==0 then 1 else x*(pow (n-1) x)} \\
   \text{poly [] y = 0} \\
   \text{poly (e:es) y = (e*(pow (length es) y)) + (poly es y)}
   \]

What is the residual code that results from partially evaluating $\text{poly}$ with respect to the input $[2, 3, 1]$, where the value for $y$ is not given? That is, compute $\text{mix} (\text{poly}, [2, 3, 1])$

9. (a) What are subtype relationship between $A$ and $B$ is required for the function
   \[ G : A \to B \]
   to have a least fixed-point? In other words, what relationship must hold between $A$ and $B$ for $\text{fix}(G)$ to be well-defined?
   (b) What is the significance of this observation, when applied to the object encodings using fixed-points and records given in Pierce?

10. In your own words, summarize the difference between the definition and use of algebras and co-algebras.

11. Apply type inference and give the (potentially polymorphic) types of $f$, $g$, and the final result. Show you work, including what constraints are solved to compute the type.

   \[
   \text{let } f = \lambda f. \lambda g. \lambda x. f(x) (g(x)) \\
   \text{g = f (\lambda a. \lambda b. a+b) (\lambda c. c * c)} \text{ in} \\
   \text{in g(5)}
   \]