FLOLAC '07
Type Systems
Exercise 1

1. Please give the *type derivations* (proof trees) for the following Mini-Haskell expressions. You should try to derive the most general type for them.
   (a) **let id = \x -> x in id id**
   (b) **\f -> f (\x -> x)**
   (c) **\x-> let f = \y -> x in (f 1, f True)**

2. (a) In Mini-Haskell+Type Classes, what would be the type inferred for the expression: **\x -> \y -> x /= (y (x, 1)).**
   (b) If you enter the above expression into Hugs using "**:t \x y -> x /= (y (x, 1))**", you will get a different type scheme, a more general one. How would you interpret this difference? Hint: consider the type for the overloaded numeric literal "1".

3. Mini-Haskell does not support recursive function definitions! One way to extend Haskell with recursive functions is to add a new form of function declaration as follows:

   E ::= …
   | letrec f = E1 in E2 --**E1** may contain a reference(s) to **f**

   For example:

   letrec fac = \x -> if x==0 then 1 else x * fac (x-1)

   Please add a typing rule for recursive function definitions. Hint: the type of **f** must be monomorphic and is the same as that of E1.