Assignment 8 - Basic Polymorphic Typechecking

1. Easy problems from CRFP
   a. Do the following pairs of types unity? If so, give a most general unifier for them. If not, explain why they fail to unify.

      (Int -> b) and (a -> Bool) - Yes - Int -> Bool
      (Int, a, a) and (a,a,[Bool]) - Yes - (Int, a, [Bool])

   b. Give the substitutions to show that the type (a, [a]) unifies with (b,c) to give (Bool, [Bool]). Bool/a, a/b, [a]/c
   c. Can the function

      f :: (a, [a]) -> b

      be applied to the following arguments. If so, what are the types of the results? Explain your answers for each.

      (2,[3]) Yes, a unifies with Int
      (2, []) Yes, a unifies with Int, and the list data type includes the null list
      (2,[True]) No, the data types for a must be the same

   d. Repeat the previous question for the function

      f :: (a, [a]) -> a
      Same answers. We're not given any results

   e. Give the type of f [] [] if f has the type

      f :: [a] -> [b] -> a -> b
      f [][] :: a -> b

   f. Explain why the following expressions do not type-check

      curry uncurry *curry's first parameter must be a function whose first parameter is a duple*
      curry curry *same reason - curry's first param is a function, not a duple*

2. (Derived from CRFP) Give the derivation of the type of the function h defined as

   h x = f x x

   I can't figure out where in CRFP you are referring. I hope this is the general:
   x :: a
   f :: a -> a -> b
   f x x :: b
   h x :: a -> b

3. (Derived from CRFP) Give the derivation of the type of each of the following expressions

curry id
curry :: ((a,b) -> c) a -> b -> c
id :: (a -> a)
or
id :: ((a,b) -> (a,b))
and
curry :: ((a,b) -> (a,b)) -> a -> b -> (a,b)
curry id :: a -> b -> (a,b)

uncurry id
uncurry :: (a -> b -> c) -> (a,b) -> c
id :: a -> a
or
id :: (b -> c) -> (b -> c)
and
uncurry :: ((b->c) -> b -> c) -> (b->c,b) -> c
uncurry id :: (b -> c,b) -> c

curry (curry id)
id :: ((a,b),c) -> ((a,b),c))
curry :: (((a,b),c),((a,b),c)) -> (a,b) -> c -> ((a,b),c)
curry id :: (a,b) -> c -> ((a,b),c)
curry (curry id) :: a -> b -> c -> ((a,b),c)

uncurry (uncurry id)
id :: (a->b->c) -> (a->b->c)
uncurry :: ((a -> b -> c -> a -> b -> c) -> (a -> b -> c, a->b) -> c
uncurry id :: (a->b->c->a,b) ->c
uncurry (uncurry id) :: ((a->b->c,a),b) -> c