MIDTERM

LASTNAME:  FIRSTNAME:  GRADE:

- You can also write on the back of the pages. If you absolutely need to turn in more and separate pages, then write your full name on every sheet that you turn in separately!
- This midterm is CLOSED BOOK (no books, private notes, laptops, etc.)
- Academic honesty is mandatory (a grade of 0 points is assigned otherwise)
- Write legible and concise explanations
- The points for each problem indicate the relative weight. You have roughly 1 minute per point.

Problem 1 (14+3+3+3, Composite Data Types)
   a) Give an example type declaration (in Haskell) for each of the following types (note: not all examples have to be different):
      - an enumeration type

      - a product type

      - a union type

      - a list type

      - a record (tuple) type

      - a recursive type

      - a non-recursive type

      - an algebraic (“sum of products”) type
b) Explain briefly the difference between a *polymorphic type* and a “normal” type (i.e., a non-polymorphic or *monomorphic* one). Hint: you may want to use an example from part (a) in your explanation.

c) What is the difference between a “regular” array (e.g., `myboringarray = array[0..100] of integer`) and an *associative array*?

d) What is the relation between arrays and finite functions?

**Problem 2 (2+3+3, Function Signatures)** Consider the following two function signatures:

```
add1 :: Integer -> Integer -> Integer
add1 x y = x + y

add2 :: (Integer, Integer) -> Integer
add2 x y = x + y
```

a) What is the *type* and the *value* of `add1 3 4`?

b) What is the *type* of `add1 -1` and what does it stand for?

c) What happens if instead of `add1 -1` we consider `add2 -1`?
Problem 3 (4+6+2 Reduction Strategies) Consider the following functions:

\[
\begin{align*}
\text{double } x &= x + x & \text{-- (d)} \\
\text{second } x y &= y & \text{-- (s)} \\
\text{f } x y &= \\
& \mid x == y = x & \text{-- (f1)} \\
& \mid x < y = f y x & \text{-- (f2)} \\
& \mid \text{otherwise } = f y (x-y) & \text{-- (f3)}
\end{align*}
\]

a) Reduce the expression double (second (double 2) (double 3)) leftmost outermost.

b) Reduce f 4 6 and f 7 12.

c) What does this function compute? What happens for f 0 3?

Problem 4 (4+15, Abstract Data Types) a) What is an abstract data type (ADT)? (Hint: name the two distinct parts of an ADT and briefly explain their role.)
b) Define an ADT IntStack (stack of integers) in Haskell. Instead of using two separate functions, use just one function pop that returns both the topmost stack element, and the reduced stack. Indicate clearly each of the parts mentioned in (a)!

Problem 5 (3+6, Higher Order Functions)  

a) Define the function map that takes a function \( f \) of type \( \text{a} \to \text{b} \), a list \( xs \) of type \([\text{a}]\) and returns the list of type \([\text{b}]\) in which \( f \) has been applied to each element of \( xs \).

b) Define the function length which returns the number of elements of a list using \( \text{foldr} \). Recall that the effect of \( \text{foldr} \) can be depicted as follows:

\[
\text{foldr} \ (\otimes) \ e \ [x_1, \ldots, x_n] = x_1 \otimes (x_2 \otimes (\cdots (x_n \otimes e) \cdots))
\]
Problem 6 (2+4+5, Trees) Consider the following Haskell function `foo`:

```haskell
data MyTree a = Leaf a | Node (MyTree a) a (MyTree a)

foo (Leaf x) = [x]
foo (Node left x right) = (foo left) ++ (foo right) ++ [x]

mytree0 = Node
  (Node
    (Leaf 1)
    10
    (Leaf 3))
  20
  (Leaf 5))
```

a) Draw `mytree0` as a tree.

b) What are the types of `foo` and `mytree0`?

c) What does `foo mytree0` evaluate to? So what is `foo` doing?