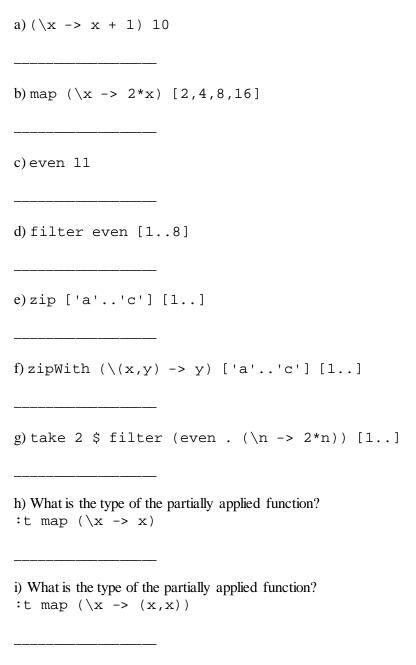
COP 4020 Programming Languages Midterm Exam on 03/03/2016 Instructor: Dr. Pawel Wocjan

Total:	/40	
Problem 3:	/10	
Problem 2:	/10	
Problem 1:	/20	
PID:		
Last name:		
First name:		

Problem 1: [20 points]

Each of the 10 subproblems is worth 2 points. Evaluate each of the expressions by hand and write down your answer in the provided space.



j) Which class constraint is missing in the definition of the function bigger? Write down the correct signature of the bigger function including the appropriate class constraint.

```
bigger :: a -> a
bigger n m = if n > m then n else m
```

Problem 2: [10 points]

a) and b) are each worth 2 points, and c) and d) are each worth 3 points.

Implement the following functions, also providing their full signatures (including class constraints when necessary). Ensure that all possible input patterns are covered by your implementation. Use the Haskell built-in function error when appropriate. Your code has to be polymorphic and not fixed to a specific data type such as Int.

The first four functions are built-in Haskell functions. Of course, you have to provide your own code.

a) head

b) tail

c) concat

```
For instance, concat [[1,2],[3],[],[4,5,6]] ~~> [1,2,3,4,5,6] concat [] ~~> []
```

d) merge

Recall that when merge receives two sorted lists as input it merges them into one sorted list

For instance,

```
merge [1,4,6] [2,3,5] ~~> [1,2,3,4,5]
merge [1,4,6] [] ~~> [1,4,6]
```

Problem 3: [10 points]

- a) is worth 2 points and b) and c) are each worth 4 points.
- a) You are given the algebraic data type

```
data Shapes = Rectangle Float Float | Circle Float
```

A rectangle is specified by its width and height and a circle by its radius. Implement the function area that computes the area of rectangles and circles. Provide the signature of the function area.

b) You are given the polymorphic algebraic data type

```
data Tree a = Nil | Node a (Tree a) (Tree a)
```

that models a binary tree.

Implement the method traverse that takes as input a binary tree and returns a list of all elements contained in the tree. The elements have to be listed in *depth-first order*, that is, always descend to the left subtree whenever possible. Otherwise backtrack and descend to a previously unexplored right subtree. Provide the full signature of the function traverse.

For instance,

```
traverse (Node 'a' (Node 'b' (Node 'c' Nil Nil) Nil) (Node 'd' Nil Nil)) ~~>
['a','b','c','d']
```

c) Using the above algebraic data type, implement a function

occurs

that takes as input a tree and an element returning whether that element occurs in the tree or not. Provide the full signature of the function occurs.

For instance,

```
occurs (Node 'a' (Node 'b' (Node 'c' Nil Nil) Nil) (Node 'd' Nil Nil)) 'c' ~~> True

occurs (Node 'a' (Node 'b' (Node 'c' Nil Nil) Nil) (Node 'd' Nil Nil)) 'e' ~~> False
```