Student Name: _____
Student ID: _____

Midterm Exam
Date: July 18, 2012
Course: COMP 348 CC 2012 /1
Instructor: Troy Taillefer

Instructions:

1. Print answers clearly do not write in cursive if your answer is illegible zero marks will be granted
2. If you write anything that is incorrect or irrelevant to the question marks will be deducted even to the point of zero.
3. Just write what you know.
4. Unless you know nothing then feel free to guess.
5. You may use Code and/or English to express your answers unless given specific directions to use one or the other.
6. When using code it is expected to compile. marks will be deducted for incorrect code
7. Answer all questions on the provided booklets.
8. When you hand in your exam you must show me your Student ID card.
9. Code quality doesn't count unless otherwise stated. Feel free to use short variable names.
10. Respect word limits don't be overly verbose.
11. Hand this exam paper and all booklets with your name on it.
12. No notes, electrical devices permitted.
13. Don't write in the grade table at the end of the exam.

1. (a) For each of the following terms give a short definition two sentences maximum.

   i. Programming Paradigm (2)

   Solution: A way of modeling and communicating computer computations and algorithms.

   ii. Type System (2)

   Solution: How and when a programming language resolves types in a programming language either at runtime (dynamic) or at compile time (static). Also if it is possible to infer types statically without declaring them.

   iii. Imperative Programming (2)

   Solution: A programming style that focuses more on how things are computed instead of what is to be computed.

   iv. Declarative Programming (2)

   Solution: A style of programming that focuses more on what is to be computed rather then how it is being computed.

   v. General Purpose Programming Language (2)

   Solution: Is a programming language that can do any possible computation that a computer(Turing complete) can do. It is used for all types of programming task.

   vi. Domain Specific Programming Language (2)

   Solution: A specialized programming language focused on a specific programming task (SQL data base queries and manipulation) often not able to do all types of computations (not Turing complete).

   (b) Using the definitions above fill in the table categorizing the following program languages. (12)

Language	Type System	Declarative/Imperative	General/Specific
C			
C++			
Haskell			
Java			
Prolog			
SQL	Static		

Solution:

Language	Type System	Declarative/Imperative	General/Specific
C	static	imperative	general
C++	static	imperative	general
Haskell	static	declarative	general
Java	static	imperative	general
Prolog	dynamic	declarative	general
SQL	Static	declarative	specific

2. (a) Briefly in a short paragraph (2-4 sentences) describe the advantages and disadvantage of Procedural versus Object Oriented programming. (4)

Solution: Procedural is good at adding new behavior to existing types, OO is good at add new types to existing behavior (protocols).

(b) Briefly 2 sentences maximum define the following terms.

i. Lambda (2)

Solution: Lambda is an anonymous function/method code block that can be created on the fly and passed around to and returned from higher order functions.

ii. Higher order function (2)

Solution: Is a function/method that takes a function and or return a function.

iii. Closure (2)

Solution: Is when a function, class, method or code block closes (inner scope) references a variable in an outer scope to configure behavior of that code block often a lambda.

iv. Monoid (2)

Solution: A container which supports being combined/composed (often folded or zipped) in some way.

v. Monad (2)

Solution: A container that represents computation and carries context (state) along with it.

3. For the following methods/functions/predicates write the most general Haskell type signature.

(a) (2)

```
answer a
  | a == 42 = True
  | otherwise = False
```

Solution:

```
answer :: Num a => a -> Bool
```

(b) (2)

```
import java.util.List;
class Lists {
    static <A extends Comparable<A>> List<A> sort(List<A> list){
        return null;
    }
}
```

Solution:

```
sort :: Ord a => [a] -> [a]
```

(c) (2)

```
length1([], 0).
length1([_|Xs], Length) :-
                  length1(Xs, NextLength),
                  Length is 1 + NextLength.
```

Solution:

```
length1 :: [a] -> Int
```

(d) (2)

```
makeMultiplier multend1 = \multend2 -> multend2 * multend1
```

Solution:

```
makeMultiplier :: Num a => a -> a -> a
```

(e) (2)

```
echoPassword _ [] = []
echoPassword f (c:cs) =
  f c : (echoPassword f cs)
```

Solution:

```
echoPassword :: (t -> a) -> [t] -> [a]
```

4. Answer the following questions about Haskell lists

   (a) What is the value of the following expression? (2)

```
take 2 [2..]
```

   Solution:

```
[2,3]
```

   (b) What is the value of the following expression? (2)

```
drop 2 [2..]
```

   Solution: tries to producing an infinite list from [5..] doesn't terminate

   (c) Write a Haskell list comprehension to generate all even numbers from zero to infinity. (3)

   Solution:

```
[2*x | x <- [1..] ]
```

5. (a) Write a Haskell interactive executable program that just echoes its inputs to its output. This program must be capable of IO redirection and piping. (3)

   Solution:

```
-- simplest
main = do
  interact id

-- less simply
main = do
     stream <- getContents
     putStrLn stream

-- least simple
main = do
     line <- getLine
     putStrLn line
     main
```

   (b) When the above program has an input from another program piped into or pipes its output into another program how is the behavior changed? (Maximum 2 sentences). (2)

   Solution: No change in behavior this is the identity executable.

   (c) When then above program is piped from or to itself how is the behavior changed? (maximum 2 sentences). (2)

   Solution: No change in behavior this is the identity executable.

6. For the following questions you may not use any higher order functions from the Haskell Prelude. You may rewrite one if know how too and want to use it.

Solution:

```
filter' _ [] = []
filter' p (a:as)
  | p a = a : rest
  | otherwise = rest
  where
    rest = filter' p as
```

(a) Write a Haskell function that acts as a high pass filter (It filters out frequencies below a certain cutoff) it will filter out all frequencies below 300 Hz the lower limits of phone voice data. This function takes a list of numbers and returns a list of numbers that are 300 or above. (8)

Solution:

```
highPass' = filter' (>=300)

highPass'' xs = [x | x <- xs, x >= 300]

highPass''' [] = []
highPass''' (x:xs)
            | x >= 300 = x : (highPass''' xs)
            | otherwise = highPass''' xs
```

(b) Write a Haskell function that acts as a low pass filter (It filters out frequencies above a certain cutoff) in this case it will filter out all frequencies above 3400 Hz the upper limit of phone voice data. This functions takes a list of numbers and returns a list of numbers that are not greater than 3400. (8)

Solution:

```
lowPass' = filter' (<=3400)

lowPass'' xs = [ x | x <- xs, x <= 3400]

lowPass''' [] = []
lowPass''' (x:xs)
           | x <= 3400 = x : (lowPass''' xs)
           | otherwise = lowPass''' xs
```

(c) Using the above two functions compose them to implement a third Haskell function that implements a band pass filter (A filter that pass frequencies in a certain range) in this case it will pass all phone voice frequencies between 300 Hz and 3400 Hz. This function takes a list of numbers and returns a list of numbers that are between 300 and 3400. (8)

Solution:

```
bandPass' = lowPass' . highPass'

bandPass'' = lowPass'' . highPass''

bandPass''' = lowPass''' . highPass'''
```

7. Prolog Monty Python Philosopher Database

(a) Write the facts in Prolog that Aristotle, Aquinas, Liebniz and Descarte are philosophers. (2)

Solution:

```
philosopher(aristotle).
philosopher(aquinas).
philosopher(liebniz).
philosopher(descarte).
```

(b) Write Prolog rule that says if you are a philosopher you drink. (2)

Solution:

```
drinks(X) :- philosopher(X).
```

(c) Write a Prolog rule that says if you drink you exist. (2)

Solution:

```
exists(X) :- drinks(X).
```

(d) Consider all the above Prolog code write the query to determine who exists and display the complete exhaustive output of the query. (2)

Solution:

```
exists(X).
X = aristotle ;
X = aquinas ;
X = liebniz ;
X = descarte ;
false.
```

(e) Write a rule that says if your a philosopher you think. (2)

Solution:

```
thinks(X) :- philosopher(X).
```

(f) Modify the exists rule to say that you exist if you think or drink. (2)

Solution:

```
exists(X) :- drinks(X).
exists(X) :- thinks(X).
```

(g) Considering your modified Prolog program write the query for who exists and display the complete exhaustive output. (2)

Solution:

```
exists(X).
X = aristotle ;
X = aquinas ;
X = liebniz ;
X = descarte ;
```

```
X = aristotle ;
X = aquinas ;
X = liebniz ;
X = descarte ;
false.
```

Question	Points	Score
1	24	
2	14	
3	10	
4	7	
5	7	
6	24	
7	14	
Total:	100	