Student Number: _____

Family Name: _____

Given Names: _____

Signature: _____

THE UNIVERSITY OF NEW SOUTH WALES

Sample Exam

2018 Session 1

COMP 3141

Software System Design and Implementation

- Time allowed: **2 hours, plus 10 minutes reading time**
- Reading time: **10 minutes**
- Total number of questions: **4 questions with a total of 14 subquestions (total of 80 marks)**
- Total number of pages: 6, page 2 is empty.
- Answer **all** questions
- The questions are **not** of equal value
- Please answer questions 1 and 2 in the first answer booklet and questions 3 and 4 in the second booklet.
- This paper may **not** be retained by the candidate
- **Answers must be written in ink**, with the exception of graphs
- Students are permitted 2 A4 pages of handwritten notes (single sided), no other materials permitted.
- There is a 3% penalty if you do not fill in your student number and name correctly

Instructions

You should have received two answer booklets. Please write the answers to Question 1 & 2 into one booklet and the answers to Question 3 & 4 into the other booklet.

Whenever the answer to a questions requires Haskell code, make your answers as clear and easy to understand as possible. Provide brief comments where necessary. Confusing and illegible solutions will lose marks. Small syntactic mistakes will not lose marks, but serious mistakes will.

Question I [20 Marks]

*Answer this question in the **first** answer booklet.*

Please answer all questions in a concise manner — a few sentences are enough for each subquestion. Overly verbose answers will lose marks.

(1) **[5 marks]**
What is the difference between a total and a partial function? Give an example of each.

Solution: A total function returns a result for every possible input value.

(2) **[5 marks]**
Give an example of a function which cannot be expressed in a turing complete programming language.

Solution: Halting problem (also, see quiz for other examples).

(3) **[5 marks]**
How can code coverage tools help to assess the quality of your tests?

Solution: With these tools, you can check whether all expression/conditionals/paths in your program have been executed. If not, there is either dead code in your program (i.e., executions which can never be executed) or the tests are not sufficient.

(4) **[5 marks]**
Program properties can be checked dynamically and statically. Give an example of both a dynamic check and a static check in either C, Java, or Haskell.

Solution: Statically, for example: scoping, types. Dynamically, for examples: assertions.

Question II [20 Marks]

*Answer this question in the **first** answer booklet.*

Please answer all questions in a concise manner — a few sentences are enough for each subquestion. Overly verbose answers will lose marks.

Consider the following definitions we can use to define a statically typed `printf` function:

```
data List a = Nil | a ::: List a

type family FormatArgsThen (fmt :: List *) (ty :: *)
type instance FormatArgsThen Nil ty = ty
type instance FormatArgsThen (t ::: fmt) ty = t -> FormatArgsThen fmt ty

printf :: Format fmt -> FormatArgsThen fmt String
```

(1) **[5 marks]**
What is the purpose of the type family? What property does it guarantee, and how does it do that?

> **Solution:** The type family statically determines the type of 'printf' given a specific format argument. Hence, we can be sure to pass the number and type of arguments that match the format string. It achieves that by traversing the type level list of format parameters and constructing an appropriate function type from that.

(2) **[5 marks]**
What type does `FormatArgsThen (Int ::: String ::: Double ::: Nil) (IO ())` evaluate to during type checking?

> **Solution:** `Int -> String -> Double -> IO ()`

(3) **[5 marks]**
What is a "pure function"? According to your definition, is

```
getChar:: IO Char
```

a pure function?

> **Solution:** A pure function is a function whose result only depends on its arguments, and which does not cause any observable side effects.
> The `getChar` function is therefore semantically a pure function, as the IO-type encapsulates a world which is passed in as argument, and returns a new world, as well as a charater.

(4) **[5 marks]**
In C, the evaluation of the expressions which have side-effectful subexpression, for example `(getchar() < getchar())`, can lead to undefined behaviour. How is this different in Haskell?

> **Solution:** In general, the evaluation order of expressions is much more defined in C than in Haskell (e.g., bindings in Haskell are not evaluated in the order they are listed), in this case, Haskell forces the programmer to commit to an evaluation order, as the two IO operations have to be sequences using `>>=` or, equivalently, the do-notation.

Question III [20 Marks]

*Answer this question in the **second** answer booklet.*

Here is a definition of sized vectors':

```
data Nat = Z | S Nat

data SNat (n :: Nat) where  -- natural numbers as singleton type
  Zero :: SNat Z
  Succ :: SNat n -> SNat (S n)

type family (+) (n :: Nat) (m :: Nat) :: Nat
type instance 'Z + m = m
type instance ('S n) + m = 'S (n + m)

type family (*) (n :: Nat) (m :: Nat) :: Nat
type instance 'Z     * m = 'Z
type instance ('S n) * m = m + (n * m)

data Vec (n :: Nat) a where
  VNil  :: Vec Z a
  VCons :: a -> Vec n a -> Vec (S n) a
```

(1) [**7 marks**]

Consider the following function

```
concat :: [[a]] -> [a]
concat []         = []
concat (xs : xss) = xs ++ concat xss
```

Give the corresponding function definition using the data type `Vec` instead of lists.

Solution: I

```
append :: Vec n a -> Vec m a -> Vec (n * m) a
append VNil ws = ws
append (VCons v vs) ws = VCons v (append vs ws)

concat :: Vec n a -> Vec m a -> Vec (n * m) a
concat VNil         = VNil
concat (VCons xs  xss) = appendV xs (concat xss)
```

(2) [**7 marks**]

Define a function

```
replicateV :: SNat n -> a -> Vec n a
```

that yields a vector of length n, such that every element of the vector is the value of type a passed as the second argument to `replicateV`.

Solution:

```
replicateV :: SNat n -> a -> Vec n a
replicateV Zero     x = VNil
replicateV (Succ n) x = VCons x (replicate n x)
```

(3) [**6 marks**]

On regular Haskell lists, the function

```
takeWhile :: (a -> Bool) -> [a] -> [a]
```

accepts a predicate function and a list as argument, and returns the longest prefix of the argument list of elements for which the predicate function evaluates to `True`:

```
Prelude> takeWhile (>10) [1..10]
[]
Prelude> takeWhile (<10) [1..10]
[1,2,3,4,5,6,7,8,9]
Prelude> takeWhile odd [1..10]
[1]
```

Define a corresponding function on vectors.

Solution: The function takeWhile cannot directly be defined on sized vectors, because it's not known at compile what the size of the result will be, because it depends on the data values of the input vector, not just the shape. It is possible, however, to have a function:

```
takeWhileV :: (a -> Bool) ->  Vec n a -> [a]
```

Question IV [20 Marks]

*Answer this question in the **second** answer booklet.*

(1) **[6 marks]**

Consider the following code snippet[1]:

```
data Message a = Message String

data PlainText
data Encrypted

send :: Message Encrypted -> IO ()
encrypt :: Message PlainText -> Message Encrypted
decrypt :: Message Encrypted -> Message PlainText
```

Which types in this definition are phantom types? What is the purpose of types `PlainText` and `Encrypted`?

Solution: The type `Message:: * -> *` is a phantom type, as its type parameter doesn't appear on the right hand side of the definition.
The types `PlainText` and `Encrypted`, which are empty, are used to statically mark message which are encrypted and those which are not, to enfore here, for example, that only encrypted messages can be sent.

(2) **[7 marks]**

According to the Curry-Howard Isomorphism, which programming concept corresponds to proof normalisation?

Solution: Program evaluation

Give a concrete example of a proof normalisation for a propositional logic formula and the corresponding object in the simply typed lambda calculus.

Solution: See slides, quizzes

(3) **[7 marks]**

What is the relevance of termination for the Curry Howard Isomorphism?

Solution: The Curry-Howard correspondence only applies for terminating functions.

[1] taken from Jakub Arnold's blog