Sample exam paper February 2015 **Functional Programming and Intelligent Algorithms**
Duration 4 hours **IE501614**

Calculators and other accessories are not permitted.
Include **all intermediate calculations** necessary to justify your answer.

Problem 1 ..... *(15%)*

(a) What are the values of the following Haskell expressions?

1. `map g [1,3,5] where g x = x ‘div‘ 2`
2. `[ x + 1 | x <- [ 2 ..  9 ], x ‘mod‘ 2 == 0 ]`
3. `map (*2) [1..3]`

(b) What do we mean by a function having *side effects*?

(c) Haskell supports *referential transparency*. Explain what that means.

(d) Explain what benefits we get from referential transparency.

(e) Exolain what extra challenges are caused by referential transparency.

Problem 2 ..... *(15%)*

A neuron N is defined by a weight vector $\vec{w} = (w_0, w_1, \ldots, w_n)$ and defines the function

$$f_N(\vec{x}) = g\left(\sum_{i=0} x_i w_i\right), \text{ where} \tag{1}$$

$$\vec{x} = (x_1, x_2, \ldots, x_n), \tag{2}$$

$$x_0 = -1, \tag{3}$$

$$g(h) = \begin{cases} 0, \text{ for } h \leq 0, \\ 1, \text{ for } h > 0. \end{cases} \tag{4}$$

This function is often called the *recall* of the neuron.

(a) Define a data type `Neuron` to represent a neuron in Haskell.

(b) The function g is known as the *activation* function. Give a Haskell definition of g as defined in (4).

(c) In multi-layer neural networks (backpropagation networks), other definitions are used instead of g. Why is (4) unsuitable as an activation function for the backpropagation algorithm?

(d) Give type declaration for a `recall` function which implements $f_N(\vec{x})$.

(e) Give a definition of the `recall` function.

Problem 3 ..... *(16%)*

Suppose we have trained a neural network for fingerprint identification. Presented with a two fingerprints, the system is asked whether the two match. Consider how to assess the reliability of this system.

(a) Explain how an experiment can be designed to test the system.

(b) Name and define at three distinct heuristics which can be used to describe the reliability of the system.

(c) The number of epochs (iterations) of training is important for the performance. How do you expect the error rates to vary as the number of epochs increase? Give reasons for your answer.

(d) How do you find the optimal number of epochs of training for a neural network?

Problem 4 ..... *(0%)*

Consider the following function:

Problem 5 ..... *(0%)*

Consider the following code:

```
1  f0, f1 :: Double → Double
2  f0 x = x^2 + 8*x + 8
3  f1 x = x^2 + 2*x + 1
4
5  fmin :: (Double → Double)              -- < 1. Your explanation here! >
6                  → Double               -- < 2. Your explanation here! >
7                  → Double               -- < 3. Your explanation here! >
8                  → (Double,             -- < 4. Your explanation here! >
9                     Double,             -- < 5. Your explanation here! >
10                    Int)                -- < 6. Your explanation here! >
11 fmin f x0 delta | x0 − delta < l || x0 + delta > u = (x0, f x0, 0)
12                 | f (x0 − delta) < f x0            = (a, b, c + 1)
13                 | f (x0 + delta) < f x0            = (a', b', c' + 1)
14                 | otherwise                        = (x0, f x0, 0)
15                    where l = −20        -- < 7. Your explanation here! >
16                          u = 20         -- < 8. Your explanation here! >
17                          (a,b,c)     = fmin f (x0 − delta) delta
18                          (a',b',c') = fmin f (x0 + delta) delta
```

(a) 1. Explain why the function $f_0(x) = x^2 + 8x + 8$ can be optimised analytically.

2. Ignoring possible values at $x = \pm\infty$, find the optimum $f_0(x_{\text{opt}})$ and determine whether it is a minimum or a maximum.

3. Plot the function by hand and indicate x_{opt} and $f_0(x_{\text{opt}})$. Remember to label your axes.

(b) Consider the function `fmin` in the code above. Example usage in the ghci interpreter is given below:

```
1  *SampleExam> fmin f0 10 0.01
2  (−3.99999999999979,−8.0,1400)
```

1. Replace the 8 placeholders in the code for `fmin` above with your own meaningful comments, that is, a short explanation of each parameter.

2. Give a high-level explanation (as if you were talking to a friend or your parents without the aid of any code or notes) of how the algorithm implemented by function `fmin` works and point out its shortcomings.

3. Below is an example of applying the algorithm on the function `f1`.

```
1  *SampleExam> fmin f1 5 1
2  (−1.0,0.0,6)
```

The algorithm iterates by recursion. Show by hand which guard that evaluates to `True` at each recursive call until the function returns. You should illustrate by doing the necessary evaluation, e.g., write
Step 0: *fmin f1 5 1 → f1 (5-1) = 25 < f1 5 = 36 → (a, b, c + 1) → fmin f1 4 1*
Step 1: *fmin f1 4 1 ...*

Problem 6 ..... *(0%)*

A genetic algorithm (GA) is typically described as having the following steps:

Step 1 Define cost function and variables coded as chromosomes.

Step 2 Choose GA parameters such as selection rate, mutation rate, elitism, stopping criteria, population size.

Step 3 Generate initial population.

Step 4 Evaluate the cost of each chromosome in the population.

Step 5 Select chromosome parents for reproduction.

Step 6 Perform mating.

Step 7 Perform mutation.

Step 8 Unless stopping criteria satisfied, repeat from Step 4.

(a) Explain the terms (i) population; (ii) chromosome; (iii) gene; and (iv) cost function.

(b) Explain the purpose of mutation and give an example of how mutation can be performed for (i) a binary GA; and (ii) a continuous GA.

Problem 7 ..... *(0%)*

Consider the code segment from a binary GA given below:

```
1  type Gene = [Int]
2  type Chromosome = [Gene]
3  type Population = [Chromosome]
4
5  g1,g2,g3,g4 :: Gene
6  c1,c2 :: Chromosome
7  p :: Population
8  g1 = [1,0,1,0,1,0]
9  g2 = [0,1,0,1,0,1]
10 g3 = [0,0,0,0,0,0]
11 g4 = [1,1,1,1,1,1]
12 c1 = [g1,g2]
13 c2 = [g3,g4]
14 p  = [c1,c2]
15
16 isEqual :: (Eq a) => a -> a -> Int
17 isEqual a b = 0 -- < 1. Replace 0 with your own code! >
18
19 geneCost :: Gene -> Gene -> Int
20 geneCost g1 g2 = sum $ zipWith isEqual g1 g2
21
22 chromCost :: Chromosome -> Chromosome -> Int
23 chromCost c1 c2 = 666 -- < 2. Replace 666 output with your own code! >
24
25 chromCostP :: Chromosome -> Chromosome -> (Int, Chromosome)
26 chromCostP target c = (sum $ zipWith (geneCost) target c, c)
27
28 evalPop :: Chromosome -> Population -> [(Int, Chromosome)]
29 evalPop target pop = [(6,c1),(0,c2)] -- < 3. Replace output with your own code! >
30
31 sortPop :: [(Int, Chromosome)] -> [(Int, Chromosome)]
32 sortPop pop = [(0,c2),(6,c1)] -- < 4. Replace output with your own code! >
```

The function

- `geneCost` compares two genes bit by bit and returns a cost equal to the total number of unequal bits.
- `chromCost` compares two chromosomes in the same fashion.
- `chromCostP` compares a chromosome with a target chromosome and returns the cost as well as the chromosome itself in a pair (tuple).
- `evalPop` evaluates all the chromosomes in a population by comparing them to a target chromosome.
- `sortPop` sorts an evaluated population in increasing order of cost.

(a) Implement `isEqual` (see comment #1).

(b) Implement `chromCost` (see comment #2).

(c) Implement `evalPop` (see comment #3).

(d) Implement `sortPop` (see comment #4).

For all implementations, please ignore the existing hardcoded implementations/answers (e.g., 666). This code has been placed there just to make the script compile. You must enter your own *generic* implementations.

Problem 8 ..... *(0%)*

The research presented in Lecture 5 on a real-world application of GAs shows that it is possible to optimise the dynamic positioning of tugs in northern Norway by means of a GA and so-called receding horizon control (RHC).

(a) Explain briefly the principles of RHC.

(b) Based on the solution from the presented research, explain how an algorithm that combines RHC with a GA can solve this dynamic tug fleet optimisation problem.