Lab Session Software Testing 2013, Week 2 With each deliverable, indicate the time spent.

1. Write a program (in Haskell) that takes a triple of integer values as arguments and gives as output one of the following statements:

   - 'Not a triangle' ('Geen driehoek') if the three numbers cannot occur as the lengths of the sides of triangle,
   - 'Equilateral' ('Gelijkzijdig') if the three numbers are the lengths of the sides of an equilateral triangle,
   - 'Rectangular' ('Rechthoekig') if the three numbers are the lengths of the sides of a rectangular triangle,
   - 'Isosceles' ('Gelijkbenig') if the three numbers are the lengths of the sides of an isosceles (but not equilateral) triangle,
   - 'Other' ('Anders') if the three numbers are the lengths of the sides of a triangle that is not equilateral, not rectangular, and not isosceles.

   Here are some useful type definitions:

   ```
   data Shape = NoTriangle | Equilateral
              | Isosceles  | Rectangular | Other deriving (Eq,Show)

   triangle :: Integer -> Integer -> Integer -> Shape
   ```

   You may wish to consult `http://en.wikipedia.org/wiki/Triangle`. Indicate how you *tested* or *checked* the correctness of the program.

   Deliverables: Haskell program, test report of 1/2 A4, indication of time spent.

2. Implement the *Joe* and *Jill* programs from the lecture notes for this week, using Haskell or your favorite programming language. Discuss how you would test that each of the two programs follows the best possible strategy. The *types* for *Joe* and *Jill* are given in the lecture notes.

   Deliverables: two programs, test report of 1/2 A4, indication of time spent.

3. The lecture notes of this week discuss the notions of satisfiability, tautology, contradiction, logical entailment and logical equivalence for formulas of propositional logic.

   The lecture notes give a definition of `satisfiable`, for objects of type `Form`.

   Your task is to give definitions of:

   ```
   contradiction :: Form -> Bool

   tautology :: Form -> Bool

   -- logical entailment
   entails :: Form -> Form -> Bool

   -- logical equivalence
   equiv :: Form -> Form -> Bool
   ```

   Use a module that imports `Week2.hs`. Check that your definitions are correct.

   Deliverables: implementation, description of your method of checking the definitions, indication of time spent.

4. The lecture notes of this week discuss the conversion of Boolean formulas (formulas of propositional logic) into CNF form. The lecture notes also give a definition of a Haskell datatype for formulas of propositional logic, using lists for conjunctions and disjunctions. Your task is to write a Haskell program for converting formulas into CNF, and to test whether the conversion is correct.

   Deliverables: conversion program, implementation of a number of tests, test report, indication of time spent.