Lab Session Software Testing 2014, Week 3 With each deliverable, indicate the time spent.

1. 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 `Week3.hs`. Check that your definitions are correct.

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

2. 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.

   Deliverables: conversion program with documentation.

3. Implement a test generator for testing the correctness of the conversion program from the previous exercise. What are relevant properties to test?

Deliverables: test generator, sequence of test properties, test report, indication of time spent.

4. In SAT solving, one common technique is resolution style theorem proving. For that, it is usual to represent a formula in CNF as a list of clauses, were a clause is a list of literals, and where a literal is represented as an integer, with negative sign indicating negation. Here are the appropriate type declarations.

   ```
   type Clause  = [Int]
   type Clauses = [Clause]
   ```

   Clauses should be read disjunctively, and clause lists conjunctively. So 5 represents the atom p_5, -5 represents the literal $\neg p_5$, the clause $[5, -6]$ represents $p_5 \vee \neg p_6$, and the clause list $[[4], [5, -6]]$ represents the formula $p_4 \wedge (p_5 \vee \neg p_6)$.

   Write a program for converting formulas to clause form. You may assume that your formulas are already in CNF. Here is the appropriate declaration:

   ```
   cnf2cls :: Form -> Clauses
   ```

   If you combine your conversion function from Exercise 2 with `cnf2cls` you have a function that can convert any formula to clause form.

   Employ automated testing to check whether your translation is correct, using some appropriate properties to check.

   Deliverables: Conversion program, test properties, documentation of the automated testing process.