CMSC 22610 Winter 2005	**Implementation of Computer Languages**	**Homework 1 Due January 11**

Consider the language of *propositional formulae* formed from variables ($a, b, c, \ldots$), negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$), according to the following abstract syntax:

$$\begin{array}{rcl} \phi & ::= & a \\ & | & \neg\phi_1 \\ & | & \phi_1 \wedge \phi_2 \\ & | & \phi_1 \vee \phi_2 \end{array}$$

We can represent propositional formulae in SML using the following datatype:

```
datatype prop
  = Var of string
  | Not of prop
  | And of prop * prop
  | Or of prop * prop
```

For example, the formula $a \wedge \neg(b \vee \neg c)$ is represented as the SML value

```
And(Var "a", Not(Or(Var "b", Not(Var "c"))))
```

We define the language of *disjunctive normal forms* as

$$\begin{array}{rcl} D & ::= & C \\ & | & A \vee D \\ C & ::= & A \\ & | & A \wedge C \\ A & ::= & a \\ & | & \neg a \end{array}$$

This language can be represented as the following SML datatype:

```
datatype dnf = Or of conjunct list
and conjunct = And of atom list
and atom = Var of string
  | Not of string
```

Because we have used the same constructor names, we must put the `prop` and `dnf` types in separate modules:

```
structure Prop =
  struct
    datatype prop = ...
  end
```

```
structure DNF =
  struct
    datatype dnf = ...
  end
```

One can convert an arbitrary formula to DNF by using the following rewrite rules:

$$\begin{aligned}
\neg(\neg\phi) &\Rightarrow \phi \\
\neg(\phi_1 \wedge \phi_2) &\Rightarrow \neg\phi_1 \vee \neg\phi_2 \\
\neg(\phi_1 \vee \phi_2) &\Rightarrow \neg\phi_1 \wedge \neg\phi_2 \\
\phi_1 \wedge (\phi_2 \vee \phi_3) &\Rightarrow (\phi_1 \wedge \phi_2) \vee (\phi_1 \wedge \phi_3) \\
(\phi_1 \vee \phi_2) \wedge \phi_3 &\Rightarrow (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3)
\end{aligned}$$

Your assignment is to write an SML function (`toDNF`) that converts propositional formulae to their equivalent DNF. It should have the following signature:

```
val toDNF : Prop.prop -> DNF.dnf
```

You solution should consist of four files: `prop.sml` (holding the module `Prop`), `dnf.sml` (holding the module `DNF`), `convert.sml` (holding the `Convert` module, which contains the `toDNF` function), and `hw1.cm` (containing the CM specification). Please ensure that your name appears in a comment at the beginning of each file.

The CM specification should be as follows:

```
Library
  structure Prop
  structure DNF
  structure Convert
is
  $/basis.cm
  prop.sml
  dnf.sml
  convert.sml
```