CMSC 22610	Computer Language Implementation	Homework 6
Winter 2005		Due March 10

1. Consider the following lexically scoped language of integer expressions:

$$
\begin{array}{rcll}
exp & ::= & NUM & (1) \\
 & | & VAR & (2) \\
 & | & \textbf{let } VAR = exp_1 \textbf{ in } exp_2 & (3) \\
 & | & exp_1 + exp_2 & (4)
\end{array}
$$

   Give an attribute grammar that computes the value of an expression. You may assume that NUM.`value` is the integer value of the numeric literal and that VAR.`name` is the name of a variable. Your solution may use functional data structures, such as sets and finite maps.

2. Consider the following representation of terms in SML:

```
datatype term = T of (string * term list)
```

   where the `string` is the operator name. It is possible to define strategy combinators for this term representation, where a strategy has the type

```
type strategy = term -> term option
```

   and `NONE` denotes failure. For example,

```
fun <+ (s1, s2) t = (case s1 t
       of NONE => s2 t
        | someT => someT
      (* end case *))
```

   implements deterministic choice and

```
fun all s (T(f, args)) = let
      fun try ([], l) = SOME(T(f, List.rev l))
        | try (t::ts, l) = (case s t
             of NONE => NONE
              | SOME t' => try(ts, t'::l)
            (* end case *))
      in
        try (args, [])
      end
```

   implements the `all` combinator.

   (a) Give the SML code for the `test` combinator. Recall that the `test` combinator acts as the identity when its argument succeeds and fails when its argument fails.

   (b) Give the SML code for a generic congruence operator with the following specification:

```
val congruence : (string * strategy list) -> strategy
```