CMSC 22610	**Computer Language Implementation**	**Homework 5**
Winter 2004		**Due March 12**

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.

1. Give the SML code for the `test` combinator.
2. Give the SML code for a generic congruence operator with the following specification:

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

3. Give the SML code for the `one` combinator.
4. How would you implement the following rewrite rule in this framework:

$$\texttt{Plus}(\texttt{Times}(a, c), \texttt{Times}(b, c)) \Longrightarrow \texttt{Times}(a, \texttt{Plus}(b, c))$$