CMSC 22610	**Computer Language Implementation**	**Homework 5**
Winter 2007		**Due February 27**

1. Recall the discussion of polymorphic typechecking in Handout 5. Assume that we have both `int` and `real` as base types. To extend the typechecker to support overloaded functions on integers and reals (*e.g.*, "+"), we need to allow type variables that are restricted to be members of some set. For example, the type of "+" could be written as

$$\forall\alpha \in \{\texttt{int}, \texttt{real}\}.(\alpha \times \alpha) \to \alpha$$

   We can model this by changing the representation of type-variable kinds:

```
and tvar_kind
  = INSTANCE of ty
  | UNIV of int
  | NUMKIND
```

   Give a modified version of the unification algorithm from Section 4 of Handout 5 that deals with this new kind representation.

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

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

   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.