Introduction to CL I (684.01) OSU Linguistics
Detmar Meurers Winter 2005

**Exercise sheet 2**
(Submit as a plain text email message to dm@ling.osu.edu before class on Tuesday, Jan. 18)

Provide Prolog definitions for the following relations. Define them using the minimal means necessary—in particular, there is no need to make use of other relations defined in class or predefined predicates.
Thoroughly test your predicates before handing them in!

1. `next_to_last/2`: a two place relation which takes a list as first argument and returns the next to last element of that list (if there is one) as second argument; i.e., `last(+`*`List`*`,−`*`Next-to-Last-List-element`*`)`

   Example queries:

   ```
   ?- next_to_last([a,b,c,d]),X). ⇒ X=c
   ?- next_to_last([a,b,c],X). ⇒ X=b
   ?- next_to_last([],X). ⇒ no
   ```

2. `wrap_with_x/2`: a two place relation which takes a list and returns the same list with the functor x wrapped around every element; i.e.,

   `wrap_with_x(+`*`List`*`,−`*`List-With-x-Wrapped-Elements`*`)`

   Example queries:

   ```
   ?- wrap_with_x([a,b,c,d],X). ⇒ [x(a),x(b),x(c),x(d)]
   ?- wrap_with_x([],X). ⇒ X=[]
   ```

3. `delete_b/2`: a two place relation which takes a list and deletes one occurrence of `b` (if there is one); i.e., `delete_b(+`*`List`*`,−`*`List-with-one-b-less`*`)`

   Example queries:

   ```
   ?- delete_b([b,e,b,d],X). ⇒ X=[e,b,d]; X=[b,e,d]
   ?- delete_b([e,b,c,b,g,h],X). ⇒ X=[e,c,b,g,h]; X=[e,b,c,g,h]
   ?- delete_b([e,g,b],X). ⇒ X=[e,g]
   ?- delete_b([e,g,b,b],X). ⇒ X=[e,g,b]; X=[e,g,b]
   ?- delete_b([e,c],X). ⇒ no
   ```

   Define the relation `delete_one_b` that removes only the first occurrence of a `b`.

   ```
   ?- delete_one_b([b,e,b,d],X). ⇒ X=[e,b,d]
   ?- delete_one_b([e,b,c,b,g,h],X). ⇒ X=[e,c,b,g,h]
   ?- delete_one_b([e,g,b],X). ⇒ X=[e,g]
   ?- delete_one_b([e,g,b,b],X). ⇒ X=[e,g,b]
   ?- delete_one_b([e,c],X). ⇒ no
   ```

4. `last_added_first/2`: a two place relation which takes a list and returns the same list with the last element of the input list added to the beginning of the result; i.e., `last_added_first(+`*`List`*`,−`*`List-With-Last-Added-First`*`)`

   Example queries:

   ```
   ?- last_added_first([a,b,c,d],X). ⇒ [d,a,b,c,d]
   ?- last_added_first([a,b,c],X). ⇒ [c,a,b,c]
   ?- last_added_first([],X). ⇒ no
   ```

5. `in_list/2`: a two place relation which succeeds if the first list is a sublist (with sublist being reflexive) of the second; i.e., `in_list(+`*`Sublist`*`,+`*`List`*`)`

   Example queries:

   ```
   ?- in_list([b,c],[a,b,c,d]).  ⇒ yes
   ?- in_list([b,c],[b,c]).  ⇒ yes
   ?- in_list([b,c],[a,b,b,c,d]).  ⇒ yes
   ?- in_list([a,b],[a,b,c,d]).  ⇒yes
   ?- in_list([a,b,c],[a,b,c,d]).⇒ yes
   ?- in_list([a],[a,b,c,d]).  ⇒ yes
   ?- in_list([],[a,b,c,d]).  ⇒ no
   ?- in_list([a,c],[a,b,c,d]).  ⇒ no
   ?- in_list([b,d],[a,b,c,d]).  ⇒ no
   ```

6. `mix/2`: a two place relation which takes a list as its first argument and returns as second argument each list that consists of all and only the elements of the input list in any order of occurrence; i.e., `mix(+`*`List`*`,−`*`Mixed-list`*`)`

   Example queries:

   ```
   ?- mix([a,b,c],X). ⇒ X = [a,b,c] ; X = [b,a,c] ; X = [b,c,a] ; X = [a,c,b]
   ; X = [c,a,b] ; X = [c,b,a]
   ```

   Hint: in defining `mix` it is useful to define an auxiliary relation `insert` which inserts a single element into an input list at any arbitrary position of the list and returns this newly constructed list.