

# Assignment 4

Due: Sunday, 08.05.2011, 23:59:59 via SVN

Submit your solution for this assignment into your group's SVN as an **Eclipse Project** named "assignment04".  
Programming exercises should be executable **Prolog files** in this project.  
Theoretical exercises should be submitted as **PDF files** in this project.

## Task 1. *Complexity of append/3 (2 Points)*

For the following definition of `append/3`, show that it has linear complexity by showing that the depth of the derivation tree of any query to `append/3` has  $n+1$  nodes, where  $n$  is the length of the list provided or constructed in the first argument:

```
append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).
append([], Ys, Ys).
```

## Task 2. *Shifting lists (2 Points)*

Implement a predicate "shift(List1, List2)" so that List2 is List1 'shifted rotationally' by one element to the left. For instance,

```
?- shift([1,2,3,4,5], L2), shift(L2, L3).
```

should produce

```
L2 = [2,3,4,5,1]
```

```
L3 = [3,4,5,1,2]
```

## Task 3. *Mapping list elements (4 Points)*

Implement a predicate "translate(DigitList, WordList)" that succeeds if DigitList is a list of digits and each element in WordList is the English word for the element at the same position of DigitList. For instance,

?- translate( [1,2,3], L2).

should produce

L2 = [one, two, three]

and

?- translate( L1, [one, two, three]).

should produce

L1 = [1,2,3]

**Tip:** Start by considering how to represent the mapping of digits to words and vice versa. To check whether a term is a digit use the following helper predicate, which is itself based on a few built-in predicates that we haven't discussed yet but whose semantics is obvious:

digit(X) :- number(X), X>=0, X<10.

#### Task 4. *Linear list (4 Points)*

Implement a predicate "linear(+NestedList, ?LinearList)" that succeeds whenever LinearList is a list whose elements may themselves be arbitrarily deeply nested lists and LinearList contains all elements of NestedList in the same order but without any nesting. For instance,

?-linear ( [1,[2,3,[a,[b],c]]], L2).

should produce

L2 = [1,2,3,a,b,c]

#### Task 5. *List manipulation: Inserting elements (5 Points)*

a) (2 Points) Implement a predicate with the signature "insert\_before(Elem, ElemNew, List, NewList)" that succeeds whenever NewList is the same as List but with ElemNew added before *each* occurrence of Elem. If Elem is not in List, the predicate should fail. For instance,

?- insert\_before(c,b,[a,c,a,c,c],NewList).

should produce

NewList = [a,b,c,a,b,c,b,c]

b) (2 Points) Document as precisely as possible in which invocation modes your insert\_before/4 predicate is supposed to work and what it is doing in each case. Think in particular about deterministic and nondeterministic modes. Document only the most general modes that make a difference. For example, if a binary predicate p behaves the same for the modes p(+,+) and p(-,+)

but differently for the mode  $p(-,-)$  and again differently for the mode  $p(+,-)$  then only document the behavior for  $p(+,+)$ ,  $p(-,-)$  and  $p(+,-)$ .

- c) (1 Point) Demonstrate the consistency of your documentation and implementation by submitting traces of test queries that illustrate the behavior in each documented invocation mode. In the case of a '?' mode, illustrate the behavior for the '-' case.