Declarative Programming, Midterm exam, 19th May 2006
Total time available: 90 minutes, total score: 60
Prolog (30 points)

In questions where the definition of a Prolog predicate is asked, all predicates found in the textbook and on the slides (both built-in and locally defined) may be used freely. Please refer to individual subquestions with their numbers and letters (e.g., 2.b)!

1. Determine the outcome of the following Prolog queries (error, failure, success)! In case of success, specify the resulting variable substitutions! All queries are fed to the system independently. (Assume that the `lists` library is loaded.) (5 points)

   (a) `Z = 1+5, \+ Z = 2*3.`
   (b) `X is 4-3, Y is X+1, Y = 3-1.`
   (c) `D = 3+E, \+ D = 2, R is D+1.`
   (d) `append([], [a,12|_], [A,_]).`
   (e) `2+4-2 = A-B.`

2. Write down the canonical form or draw the tree form of the both left and right hand sides of the following unifications. Specify the variable substitutions which the unifications lead to. (9 points)

   (a) `[X,[3*_]|Z] = .(Y,[[Y*2]]).`
   (b) `f(_+A*a,[C,_|B],F) = f(F+C,[3*_,b],6).`

3. Assume that the following program is loaded into the Prolog system.

```
p([A,B|_], T, E) :-
   A > T,
   A < B,
   E = A.
p([A|As], _, E):-
   p(As, A, E).
```

   Determine the values that `X` will take as a result of the following (independent) queries! Write down *all* solutions separated by semicolons, in the same order as the system would enumerate them! If there are no solutions, write `{no}`!

   (a) `p([2,4,1.2,3], 0, A).`
   (b) `p([1,5,10], 2, A).`
   (c) `p([3,4,1,5,8], 1, A).`
   (d) `p([3,4,2,5,3,2], 4, A).`
   (e) `p([2,4,6,7,8,2,4,5],3, A).`

   Consider the following procedure, which uses the `p/3` predicate defined above:

```
% p(L, Z): Z is a member of the L list such that...
p(L, Z) :- L = [A|As], p(As, A, Z).
```

   (f) Describe in a declarative manner what this `p/2` predicate does by completing the above head comment. Make sure to specify the enumeration order of the solutions! (8 points)

4. Consider a list consisting of `X-Y` pairs. We call the pairs `A-B` and `C-D` *mergeable*, if either $A = C$ or $B = D$ holds, and they can be *merged* into the pair `X-Y`, where $X = A + C$ and $Y = B + D$. Write a Prolog procedure called `merged` which, given a list of such pairs as input in its first argument, **enumerates** the merged value of all mergeable neighboring list elements, preserving the order in which they appear in the list. Enumerate each merged value exactly once! If an auxiliary procedure is deemed necessary, write a declarative head comment for it! (8 points)

```
% merged(XYs, M): M is a merged value of two neighboring pairs in XYs.
| ?- merged([1-2], M).                 ----> no
| ?- merged([1-2, 1-3], M).            ----> M = 2-5; no
| ?- merged([1-2, 1-2], M).            ----> M = 2-4; no
| ?- merged([1-2, 1-3, 4-3], M).       ----> M = 2-5; M = 5-6; no
| ?- merged([1-2, 1-2, 1-2], M).       ----> M = 2-4; M = 2-4; no
| ?- merged([1-2, 1-3, 3-5, 4-5], M).  ----> M = 2-5; M = 7-10; no
```

Declarative Programming, Midterm exam, 19th May 2006
Total time available: 90 minutes, total score: 60
Standard ML (30 points)

In questions where the declaration of an SML function is asked, all SML functions found in the textbook and on the slides (both built-in and locally defined) may be used freely. Please refer to individual subquestions with their numbers and letters (e.g., 6.b)!

The types of the built-in functions appearing in the questions (with the exception of arithmetical and relational functions):

```
List.filter  : ('a -> bool) -> 'a list -> 'a list
map          : ('a -> 'b) -> 'a list -> 'b list
foldr        : ('a * 'b -> 'b) -> 'b -> 'a list -> 'b
op@          : 'a list * 'a list -> 'a list
op::         : 'a * 'a list -> 'a list
op o         : ('a -> 'b) * ('c -> 'a) -> 'c -> 'b
op^          : string * string -> string
```

```
explode      : string -> char list
str          : char -> string
ord          : char -> int
chr          : int -> char
Char.isUpper : char -> bool
rev          : 'a list -> 'a list
not          : bool -> bool
```

5. There are exactly **two semantic errors** in each of the following (independent) syntactically correct SML expressions. Which are these? (7 points)

   (a) `(#"a"::#"b" = explode "ab", (1, 2) < (2, 1), 1 < 2 < 3)`
   (b) `[3+3, chr 93.0, 7] = [3*2, ord #"b", 0-3-4, 0]`
   (c) `map (op +) [65, 6+5, ord chr 65]`

6. What is the **value** of `x` after evaluating the following (independent) declarations? (7 points)

   (a) `val (_::_::_::x) = rev(explode "PL" @ [#"S", #"M", #"L"])`
   (b) `val (_::x::_) = List.filter (not o Char.isUpper) (explode "aBcDeF")`
   (c) `val x = #1(foldr (fn (x, (y, b)) => (x+y, b andalso x < y)) (0, true) [3,2,1])`

7. Consider the following function definitions! (7 points)

```
(* val f1 = fn : string list * string list -> string list -> string list
   val f2 = fn : string list * string list -> string list *)
fun f1 (m::ms, n::ns) rs = f1 (ms, ns) (m^n::rs)
  | f1 _ rs = rs
and f2 msns = f1 msns []
```

   (a) Show the evaluation steps of `f2 (["SM","Pro"],["L","log"])`, using the substitutuon model and eager evaluation!
   (b) What is the **value** of `x` after evaluating the following (independent) declarations?
      i. `x = f2 ([],[])`
      ii. `x = f2 (["a","b"],["c"])`
      iii. `x = f1 (["L"],["I","J"]) ["SP"]`
      iv. `x = f1 (["Er"],["la","nguage"]) (f2 (["n"],["g"]))`

8. Consider the following exception declaration:

```
exception notfound;
```

   A word (`char list`) is called a TLA (Three Letter Abbreviation), if it consists of 3 capital letters. Write an SML function called `firstTLA` which returns the first TLA found in its argument of type `char list list`, and throws a `notfound` exception, if the list doesn't contain TLA's. You may define auxiliary functions only with appropriate head-comment! (9 points)

```
(* firstTLA : char list list -> char list
   firstTLA l = the first TLA found in l *)
*)
```

Examples:
```
firstTLA [] -> ! Uncaught exception: notfound
firstTLA [[#"a"]] -> ! Uncaught exception: notfound
firstTLA [[#"a"],[#"A",#"B",#"C"]] = [#"A", #"B", #"C"]
firstTLA (map explode ["LISP","SML","Prolog"]) = [#"S", #"M", #"L"]
```