Declarative Programming, supplementary midterm exam, Budapest, 3rd May 2005, 17:15
Total time available: 90 minutes, total score: 60
Standard ML, Group „A" (30 point)

When the task is to write a function, all standard functions of SML and the functions defined in the lectures can be used. The types of the standard functions which appear in the tasks are the following:

`List.filter`	`: ('a -> bool) -> 'a list -> 'a list`	`explode`	`: string -> char list`
`foldl`	`: ('a * 'b -> 'b) -> 'b -> 'a list -> 'b`	`implode`	`: char list -> string`
`map`	`: ('a -> 'b) -> 'a list -> 'b list`	`length`	`: 'a list -> int`
`op@`	`: 'a list * 'a list -> 'a list`	`ord`	`: char -> int`
`op::`	`: 'a * 'a list -> 'a list`	`rev`	`: 'a list -> 'a list`
`op^`	`: string * string -> string`	`tl`	`: 'a list -> 'a list`

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

   (a) `[(1.3 = 2), op^("a", #"b") = "ab", [] = [4*1]]`
   (b) `(ord "B", 2-4 = 4-2, ~3.4) = (65, true, ~3-4)`
   (c) `foldl (fn (a,b) => explode a @ b) #" " ["one", "two", #"3"]`

6. What is the value of `t` after evaluating the following (independent) value-definitions? (7 points)

   (a) `val (_::_::t::_) = explode "ab" @ tl(rev(explode "cde"))`
   (b) `val (_::t) = List.filter (fn (a,b) => (a<=b)) [(4+0,2*2), (2,2-1), (2-1,2)]`
   (c) `val t = map length [explode "1a2b3c4d", [#"Q"], [], explode ""]`

7. Assume the following function definitions. (8 points)

```
(* g : int list -> int -> int list       f : int list * int list -> int list *)
fun g n xs = let fun f (a::b::c::cs, zs) =
                       if a+b>n then f(b::c::cs, 10*n+c::zs) else f(b::c::cs, zs)
                   | f (_, zs) = rev zs
             in f(xs, []) end;
```

   What is the value of `x` after evaluating the following (independent) value-definitions?

   (a) `val x = g 7 [1,2,3,4,5,6]`
   (b) `val x = g 9 [1,2,3,4,5,6]`
   (c) `val x = g 4 [1,~2,3,4,~5,6,7,8,9]`
   (d) `val x = g 9 [1,~2,3,4,~5,6,7,8,9]`

   Complete the incomplete head-comment.

   (e) `(* g 0 xs = is the list of the elements of xs which ... *)`

8. Assume the following datatype-declaration. (8 points)

```
datatype 'a H = A of 'a | B of 'a H list
```

   An `(a,b,c,d)` 4-tuple is called heavy-ended if `a + b + c <= d`. Write a function `heavyended` which, when applied to an argument of type `(int*int*int*int) H`, it returns the list of heavy-ended 4-tuples found in the argument, preserving their original order. Try to make your solution efficient and prefer the use of higher-order functions. You can use auxiliary functions if you write proper head-comment for them.

```
(*heavyended : (int * int * int * int) H -> (int * int * int * int) list
  heavyended t = the list of heavy-ended 4-tuples found in t in their original order*)
```

Examples:
```
heavyended(A(6,4,~3,3)) = [];
heavyended(A(4,3,0,8)) = [(4,3,0,8)];
heavyended(A(4,3,~7,0)) = [(4,3,~7,0)];
heavyended(B[]) = [];
heavyended(B[B[],B[],A(6,4,~2,9)]) = [(6,4,~2,9)];
heavyended(B[B[A(1,2,4,8),A(6,3,0,9),B[A(0,1,3,2),B[A(8,~7,0,0)]]],
             B[],A(4,3,1,9)]) = [(1,2,4,8), (6,3,0,9), (4,3,1,9)];
```