```
                Deklaratív programozás pótzárthelyi
                    Budapest, 2005. november 24.
                    ============================
             SML megoldások, V1.0, dp05s−zh2−mlmegol.txt

−−−−−−−−−−−−−−−−−−
   A csoport
−−−−−−−−−−−−−−−−−−

5. Az alábbi, egymástól független, szintaktikailag helyes SML−kifejezésekben
   kifejezésenként két−két statikus szemantikai hiba van.  Melyek ezek?

   (a) [op>=(2, "2" ), false = false, "c" <> #"b"]
       − int és string nem hasonlítható össze
       − string és char nem hasonlítható össze

   (b) (chr 127, 8+8 ^ 4*4, ~1) = (#"B", "", true)
       − ^ stringek összefûzésére való
       − int és bool nem hasonlítható össze

   (c) foldl (op mod) ~1.0 [(0,1),(4,3),(1,2)]
       − a második argumentumnak itt int−nek kellene lennie
       − a listának int list−nek kellene lennie
       (esetleg − mod helyett más operátornak kellene lennie,
                  pl. (fn ((x,y),z) => x*y+z))

Pontozás (összesen max. 7 pont):
   5.a − 5.c  Helyes válasz 2−2−3 pont.
   Minden hiba megtalálása 1 pontot ér, kivéve az 5.c−t, ahol a nem
   megfelelõ operátor használatának felismerésére 2 pontot adunk.

6. Mi a t értéke és típusa az alábbi, egymástól független deklarációk
   kiértékelése után?

   (a) val (_::_::t::_) = List.revAppend(explode "MaTeX", explode "" @ [#"+"])
          t = #"T" : char

   (b) val (_::_::t) = List.filter Char.isUpper (explode "LaTeX−MaTeX")
          t = [#"X", #"M",#"T", #"X"] : char list

   (c) val t = map (fn (a,b) => a−b) [(4+0,2*2), (1,2), (size "ab", size "bc")]
          t = [0, ~1, 0] : int list

Pontozás (összesen max. 7 pont):
   6.a − 6.c  Helyes válasz (1+1)−(1+1)−(2+1) pont −− (érték+típus)

7. Legyen

   fun redl f c e 0 = e | redl f c e n = redl f c (f(c,e,n)) (n−1)

   Mi az x értéke az alábbi, egymástól független deklarációk kiértékelése
   után?

     (a) val x = redl (fn (c, i, k) => i*c+2+k) 2 0 0
            x = 0

     (b) val x = redl (fn (c, _, k) => c*2+k) 2 1 1
            x = 5

     (c) val x = map (redl (fn (c, i, _) => c+i*i) 5 0) [0,1]
            x = [0,5]
```



```
     (d) val x = List.filter (fn i => i mod 5 <> 0)
                      (map (redl (fn (c, _, k) => c+k) 5 0) [0,1])
            x = [6]

Pontozás (összesen max. 8 pont):
   7.a−ra helyes válaszért 1, 7.b−7.c−re 2−2, 7.d−re 3 pont jár.

8. Leszállónak nevezzük az olyan a,b,c számhármast, amelyre
   a−b>b−c, leszállási meredekségének pedig az (a−b)−(b−c)
   különbséget.

   Írjon olyan függvényt lemer néven, amelynek egészlista az argumentuma,
   az eredménye pedig az argumentumlista egymást követõ a,b és c elemeiből
   álló leszálló számhármasok leszállási meredekségének az eredeti sorrendet
   megõrzõ listája

   Ha tud, alkalmazzon jobbrekurziót. Segédfüggvényt definiálhat, ha ír hozzá
   fejkommentet.

   (* lemer: int list −> int list
      lemer ns = az ns lista egymást követõ elemeibõl álló leszálló
      számhármasok leszállási meredekségeinek az eredeti sorrendet megõrzõ
      listája
   *)

   Példák:

   lemer [] = [];
   lemer [6,4,3] = [1];
   lemer [6,4,2] = [];
   lemer [9,5,2,1,8,4,3] = [1,2,8,3];

   Egy hatékony megvalósítása:

   fun lemer xs=
         let
             fun lm (a::b::c::xs, zs) =
                     if a−b > b−c
                     then lm(b::c::xs, ((a−b)−(b−c))::zs)
                     else lm(b::c::xs, zs)
               | lm (_, zs) = rev zs
         in
                   lm(xs, [])
         end;

Pontozás (összesen max. 8 pont):
   Minden kisebb hibáért 1−1 pont, minden súlyos hibáért 2 vagy 3 pont
   levonás.  Súlyos hibának számít pl. egy else ág elhagyása (−2 pont) vagy
   a végtelen rekurzió (−3 pont).  Segédfüggvényben a fejkomment hiánya: −2
   pont, de nem követelmény a fejkommentben a függvény típusának
   specifikálása.  Ha nagyon rossz a program hatékonysága, −2 pont a
   levonás.
```

```
                Deklaratív programozás pótzárthelyi
                    Budapest, 2005. november 24.
                    ============================
             SML megoldások, V1.0, dp05s−zh2−mlmegol.txt

−−−−−−−−−−−−−−−−−−
   B csoport
−−−−−−−−−−−−−−−−−−

5. Az alábbi, egymástól független, szintaktikailag helyes SML−kifejezésekben
   kifejezésenként két−két statikus szemantikai hiba van.  Melyek ezek?

   (a) [1.0 / 2, 171717, ord #"a" mod 3.0]
       − az osztás −− (op/) −− nincs értelmezve real és int között
       − mod nincs értelmezve int és real között

   (b) (chr 64, 4 > 4, ~12) = ("b", 2+2 = 4 div 1, ~(7−7.7))
       − chr 64 karakter, "b" string, de meg kellene egyezzen a típusuk
         (pl. "b" −> #"b" már karakter)
       − a kivonás −− (op−) −− nincs értelmezve int és real között

   (c) foldr (op/) ~10.0 [[1.4,4.4], [7.2, 3.4], [2.1, 1.2]];
       − az egységelemnek is real−nek kell lennie
       − int list−re van szükség, nem int list list−re

Pontozás (összesen max. 7 pont):
   5.a − 5.c  Helyes válasz 2−2−3 pont.
   Minden hiba megtalálása 1 pontot ér, kivéve az 5.c−t, ahol a nem
   megfelelñ lista típussal

6. Mi a t értéke és típusa az alábbi, egymástól független deklarációk
   kiértékelése után?

   (a) val (_::t::_::_) = List.revAppend(explode "" @ [#"−"], explode "MaTeX")
         t = #"M" : char

   (b) val (_::_::_::t) = List.filter Char.isLower (explode "MaTeX−LaTeX")
         t = [#"e"] : char list

   (c) val t = map (fn (a,b) => b−a) [(2,1), (2+2,4*2), (ord #"B",ord #"A")]
         t = [~1,4,~1] : int list

Pontozás (összesen max. 7 pont):
   6.a − 6.c  Helyes válasz (1+1)−(1+1)−(2+1) pont −− (érték+típus)

7.  A redr függvényt így definiáljuk:

   (* redr : ('a * 'b * int −> 'b) −> 'a −> 'b −> int −> 'b *)

   fun redr f c e 0 = e | redr f c e n = f(c, redr f c e (n−1), n)

   Mi az x értéke az alábbi, egymástól független deklarációk kiértékelése
   után?


     (a) val x = redr (fn (_, i, k) => i + 3 + k) 2 0 0
           x = 0

     (b) val x = redr (fn (c, _, k) => c * 3 + k) 2 1 1
           x = 7

     (c) val x = map (redr (fn (c, i, _) => c + i * i) 5 0) [1,0]
```



```
           x = [5,0]

     (d) val x = List.filter (fn i => i mod 3 <> 0)
                             (map (redr (fn (c, _, k) => c + k) 3 0) [1,0]);
           x = [4]

Pontozás (összesen max. 8 pont):
   7.a−ra helyes válaszért 1, 7.b−7.c−re 2−2, 7.d−re 3 pont jár.

8. Felszállónak nevezzük az olyan a,b,c számhármast, amelyre
   a−b<b−c, elszállási meredekségének pedig az (b−c)−(a−b)
   különbséget.

   Írjon olyan függvényt elmer néven, amelynek egészlista az argumentuma,
   az eredménye pedig az argumentumlista egymást követõ a,b és c elemeibõl
   álló leszálló számhármasok elszállási meredekségének az eredeti sorrendet
   megõrzõ listája

   Ha tud, alkalmazzon jobbrekurziót. Segédfüggvényt definiálhat, ha ír hozzá
   fejkommentet.

   (* elmer: int list −> int list
      elmer ns = az ns lista egymást követõ elemeibõl álló elszálló
      számhármasok elszállási meredekségeinek az eredeti sorrendet megõrzõ
      listája
   *)

   Példák:

   elmer [] = [];
   elmer [5,4,2] = [1];
   elmer [5,4,3] = [];
   elmer [9,6,2,9,3,7,8] = [1, 13, 3];

   Egy triviális, meg egy akumulátoros megvalósítás:

   fun elmer (a::b::c::ds) =
         if a−b < b−c
         then (b−c)−(a−b)::elmer(b::c::ds)
         else elmer(b::c::ds)
     | elmer _ = []

   fun elmer ys =
       let
           fun em (a::b::c::ds, xs) =
                 if a−b < b−c
                 then em(b::c::ds, ((b−c)−(a−b))::xs)
                 else em(b::c::ds, xs)
             | em (_, xs) = rev xs
       in
           em(ys, [])
       end

Pontozás (összesen max. 8 pont):
   Minden kisebb hibáért 1−1 pont, minden súlyos hibáért 2 vagy 3 pont
   levonás.  Súlyos hibának számít pl. egy else ág elhagyása (−2 pont) vagy
   a végtelen rekurzió (−3 pont).  Segédfüggvényben a fejkomment hiánya: −2
   pont, de nem követelmény a fejkommentben a függvény típusának
   specifikálása.  Ha nagyon rossz a program hatékonysága, −2 pont a
   levonás.
```