Deklaratív programozás, nagyzárthelyi, 2000. november 28. 16.15–17.45
Munkaidő: 90 perc, összpontszám: 60
Prolog, „A" csoport (30 pont)

A Prolog eljárás megírását kérő feladatokban a jegyzetben szereplő összes Prolog eljárás (akár beépített, akár a jegyzetben definiált) szabadon használható. Ha segédeljárást definiál, feltétlenül adjon meg hozzá fejkommentet; enélkül a megoldás nem értékelhető, vagy csak csökkentett pontszámot kaphat. Törekedjék egyszerű, tömör és hatékony programozásra; használjon jobbrekurziót, kerülje a felesleges hívásokat (pl. az `append` eljárás felesleges használatát)!

1. Írja fel az alábbi kifejezések alapstruktúra-alakját (azaz szintaktikus édesítőszerek nélküli formáját), vagy rajzolja fel a hozzájuk tartozó fastruktúrákat! Állapítsa meg, hogy a két kifejezés egyesíthető-e, és ha igen, milyen behelyettesítéssel! (3 pont)

```
[1-A*B, [B|A]]                 .(X, [[2]|Y])
```

2. Egy célsorozat *keresési fája* egy olyan irányított gráf, amelynek csúcsaiban célsorozatok vannak és két csúcs között akkor megy él, ha a kiinduló csúcs célsorozatából egyetlen (Prolog) redukciós lépéssel eljuthatunk a célcsúcs célsorozatába. Az élek a redukció során alkalmazott klóz sorszámával vannak címkézve. A fa gyökerében a teljes kiindulási célsorozat van, az üres célsorozatot egy négyzettel jelöljük.

   Rajzolja fel az alábbi célsorozat keresési fáját! A fa nem üres célsorozatot tartalmazó leveleinél jelezze a visszalépést! (A `delete` és a `write` szavakat rövidítheti (`d` ill. `w`).) (7 pont)

```
| ?- delete([b,y,a,f], A, [b,y,B]), write(A/B).
```

   A `delete/3` eljárás definíciója:

```
(1) delete([H|T], H, T).
(2) delete([H|T], X, [H|L]) :-
            delete(T, X, L).
```

3. Írjon olyan Prolog eljárást, amely felsorolja egy számlista azon elemeit, amelyek megegyeznek az őket követő összes szám összegével! (7 pont)

```
% követő_összeg(L, E): Az L lista E elemét L-ben követő számok összege E.

| ?- követő_összeg([10,4,3,1,2,0], E).  E = 10 ? ;  E = 3 ? ;  E = 0 ? ;  no
```

4. Írjon olyan Prolog eljárást, amely előállítja egy lista jelölt részét! Egy J lista *kijelöli* egy L lista egy részlistáját, ha L és J ugyanolyan hosszú és J csupa 0 illetve 1 elemet tartalmaz. A *kijelölt rész* az a lista, amelyben pontosan azon $L[i]$ elemek szerepelnek, amelyre $J[i] = 1$. (13 pont)

```
% jelölt_része(J, L, M): Az L lista J által kijelölt része M.

| ?- jelölt_része([0,1,1,0], [a,b,c,d], L).    L = [b,c] ? ;  no
```

Deklaratív programozás, nagyzárthelyi, 2000. november 28. 16.15–17.45
Munkaidő: 90 perc, összpontszám: 60
Standard ML, „A” csoport (30 pont)

Az SML-függvény megírását kérő feladatokban a jegyzetben szereplő összes SML-függvény (akár beépített, akár a jegyzetben definiált) szabadon használható. Ha segédfüggvényt definiál, feltétlenül adjon meg hozzá (deklaratív) fejkommentet, azaz specifikálja a függvény típusát és írja le a függvény argumentumai és az eredménye közötti kapcsolatot; enélkül a megoldás nem értékelhető, vagy csak csökkentett pontszámot kaphat. Törekedjék egyszerű, tömör és hatékony programozásra; használjon jobbrekurziót, kerülje a felesleges hívásokat.

5. Mi a típusa az alábbi `v1`, `v2`, `v3`, `v4` SML-értékeknek? Az a)–d) deklarációk egymástól függetlenek. (5 pont)

```
List.partition : ('a -> bool) -> 'a list -> ('a list * 'a list)
Char.isLower   : char -> bool
List.map       : ('a -> 'b) -> 'a list -> 'b list
^              : string * string -> string
real           : int -> real
```

a) `val v1 = ("4", #"4", 4, real 4)`
b) `val v2 = List.partition Char.isLower [#"A", #"b", #"C", #"a", #"B", #"c"]`
c) `fun v3 (g, h, i) = g(h i)`
d) `fun v4 ls = List.map op^ ls`

6. Mit ad eredményül az alábbi `f` függvény, ha (`ééhhnn` alakban) hatjegyű számként megadott születésnapjára alkalmazza? (Az `Int` modul be van töltve.) Válaszát indokolja! (5 pont)

```
ord          : char -> int
map          : ('a -> 'b) -> 'a list -> 'b list
explode      : string -> char list
Int.toString : int -> string
rev          : 'a list -> 'a list

(* f : int -> ??? *)
local
    fun f0 x = ord x - ord #"0"
    fun f1 (x::xs, y::ys, n) = f1(xs, ys, x*y+n)
      | f1 (_, ys, n) = n
in
    fun f k = let val xs = map f0 (explode(Int.toString k)) in f1(xs, rev xs, 0) end
end
```

7. Írjon olyan SML-függvényt `parosbetuk` néven, amely adott mondatbeli szavak páros sorszámú betűinek listáját adja eredményül! A *mondat* szavak, a *szó* betűk sorozata; minden szó első betűjének sorszáma 1. A szavak között pontosan egy szóköz van. A mondatban csak szóközök és betűk fordulhatnak elő, ezért minden nem szóköz karakterről feltételezheti, hogy az betű. A `Char.isSpace : char -> bool` tesztelő függvény a szóközök felismerésére használható. (9 pont)

```
(* parosbetuk t = a t-beli szavak páros sorszámú betűinek listája
   parosbetuk : string -> char list
*)
```

Példák:
```
parosbetuk "a b c d e f" = []
parosbetuk "Szent kert bo lomb" = [#"z", #"n", #"e", #"t", #"o", #"o", #"b"]
```

8. Írjon olyan SML-függvényt `fennsikok` néven, amely megszámlálja, hogy egy egészlistában hány fennsík van! *Fennsíknak* nevezzük az olyan maximális hosszúságú (se balra, se jobbra ki nem terjeszthető), legalább 3 és legfeljebb 5 hosszúságú számsorozatot, amelyben az elemek értéke azonos. (11 pont)

```
(* fennsikok ns = a legalább 3 és legfeljebb 5 hosszúságú fennsíkok
                  száma az ns listában
   fennsikok : int list -> int
*)
```

Példák:
```
fennsikok [1, 2,2,2,2, 3,3, 8,8,8,8,8,8, 7,6,7,7] = 2;
fennsikok [5,5,5, 2,2,5,5,5,5,5,5] = 1;
fennsikok [] = 0;
```

Deklaratív programozás, nagyzárthelyi, 2000. november 28. 16.15–17.45
Munkaidő: 90 perc, összpontszám: 60
Prolog, „B" csoport (30 pont)

A Prolog eljárás megírását kérő feladatokban a jegyzetben szereplő összes Prolog eljárás (akár beépített, akár a jegyzetben definiált) szabadon használható. Ha segédeljárást definiál, feltétlenül adjon meg hozzá fejkommentet; enélkül a megoldás nem értékelhető, vagy csak csökkentett pontszámot kaphat. Törekedjék egyszerű, tömör és hatékony programozásra; használjon jobbrekurziót, kerülje a felesleges hívásokat (pl. az `append` eljárás felesleges használatát)!

1. Írja fel az alábbi kifejezések alapstruktúra-alakját (azaz szintaktikus édesítőszerek nélküli formáját), vagy rajzolja fel a hozzájuk tartozó fastruktúrákat! Állapítsa meg, hogy a két kifejezés egyesíthető-e, és ha igen, milyen behelyettesítéssel! (3 pont)

   ```
   f(A/B+C, A+C, [A|B])              f(X+Y, _+2, [Y,1])
   ```

2. Egy célsorozat *keresési fája* egy olyan irányított gráf, amelynek csúcsaiban célsorozatok vannak és két csúcs között akkor megy él, ha a kiinduló csúcs célsorozatából egyetlen (Prolog) redukciós lépéssel eljuthatunk a célcsúcs célsorozatába. Az élek a redukció során alkalmazott klóz sorszámával vannak címkézve. A fa gyökerében a teljes kiindulási célsorozat van, az üres célsorozatot egy négyzettel jelöljük.

   Rajzolja fel az alábbi célsorozat keresési fáját! A fa nem üres célsorozatot tartalmazó leveleinél jelezze a visszalépést! (A `delete` és a `write` szavakat rövidítheti (`d` ill. `w`).) (7 pont)

   ```
   | ?- delete([7,2], A, B), write(A+B).
   ```

   A `delete/3` eljárás definíciója:

   ```
   (1) delete([H|T], H, T).
   (2) delete([H|T], X, [H|L]) :-
           delete(T, X, L).
   ```

3. Írjon olyan Prolog eljárást, amely felsorolja egy számlista azon elemeit, amelyek megegyeznek az őket megelőző összes szám összegével! (7 pont)

   ```
   % előző_összeg(L, E): Az L lista E elemét L-ben megelőző számok összege E.

   | ?- előző_összeg([0,2,1,3,4,10], E).  E = 0 ? ;  E = 3 ? ;  E = 10 ? ;  no
   ```

4. Írjon olyan Prolog eljárást, amely megadja, hogy két egyforma hosszú listában mely pozíciókon vannak azonos és melyeken különböző elemek! Egy J lista *jelöli* az L és a K lista *azonosságát*, ha mindhárom hossza megegyezik, és $J[i] = 1$, ha $L[i] = K[i]$, 0 egyébként. (13 pont)

   ```
   % azonos_hely(L, K, J): J lista az L és a K lista azonosságát jelöli.

   | ?- azonos_hely([a,b,c,d], [f,b,d,d], J).   J = [0,1,0,1] ? ;  no
   ```

Deklaratív programozás, nagyzárthelyi, 2000. november 28. 16.15–17.45
Munkaidő: 90 perc, összpontszám: 60
Standard ML, „B” csoport (30 pont)

Az SML-függvény megírását kérő feladatokban a jegyzetben szereplő összes SML-függvény (akár beépített, akár a jegyzetben definiált) szabadon használható. Ha segédfüggvényt definiál, feltétlenül adjon meg hozzá (deklaratív) fejkommentet, azaz specifikálja a függvény típusát és írja le a függvény argumentumai és az eredménye közötti kapcsolatot; enélkül a megoldás nem értékelhető, vagy csak csökkentett pontszámot kaphat. Törekedjék egyszerű, tömör és hatékony programozásra; használjon jobbrekurziót, kerülje a felesleges hívásokat.

5. Mi a típusa az alábbi `v1`, `v2`, `v3`, `v4` SML-értékeknek? Az a)–d) deklarációk egymástól függetlenek. (5 pont)

```
List.partition : ('a -> bool) -> 'a list -> ('a list * 'a list)
Char.isUpper   : char -> bool
List.map       : ('a -> 'b) -> 'a list -> 'b list
@              : 'a list * 'a list -> 'a list
round          : real -> int
```

a) `val v1 = (4.0, "4.0", #"4", round 4.0)`
b) `val v2 = List.partition Char.isUpper [#"a", #"b", #"C", #"A", #"B", #"c"]`
c) `fun v3 (g, h, i) = (g h) i`
d) `val v4 = List.map op@`

6. Mit ad eredményül az alábbi `f` függvény, ha (ééhhnn alakban) hatjegyű számként megadott születésnapjára alkalmazza? (Az `Int` modul be van töltve.) Válaszát indokolja! (5 pont)

```
ord          : char -> int
map          : ('a -> 'b) -> 'a list -> 'b list
explode      : string -> char list
Int.toString : int -> string
rev          : 'a list -> 'a list

(* f : int -> ??? *)
local
    fun f0 x = ord #"5" - ord x
    fun f1 (k, x::xs, y::ys) = f1((x*y)+k, xs, ys)
      | f1 (k, xs, _) = k
in
    fun f k = let val xs = map f0 (explode(Int.toString k)) in f1(0, rev xs, xs) end
end
```

7. Írjon olyan SML-függvényt `kozepbetuk` néven, amely adott mondatbeli páratlan hosszú szavak középső betűiből álló listát ad eredményül! A *mondat* szavak, a *szó* betűk sorozata. A szavak között pontosan egy szóköz van. A mondatban csak szóközök és betűk fordulhatnak elő, ezért minden nem szóköz karakterről feltételezheti, hogy az betű. A `Char.isSpace : char -> bool` tesztelő függvény a szóközök felismerésére, a `String.sub : string * int -> char` függvény pedig a 0-tól indexelt füzér adott indexű karakterének kiválasztására használható. (9 pont)

```
(* kozepbetuk t = a t-beli páratlan hosszú szavak középső betűinek listája
   kozepbetuk : string -> char list
*)
```

Példák:
```
kozepbetuk "a b c d e f" = [#"a", #"b", #"c", #"d", #"e", #"f"]
kozepbetuk "s ont monoton but konokon es fajon" =  [#"s", #"n", #"o", #"u", #"o", #"j"]
kozepbetuk "Szent kert bo lomb" = [#"e"]
```

8. Írjon olyan SML-függvényt `dimbdombok` néven, amely megszámlálja, hogy egy egészlistában hány dimbdomb van! *Dimbdombnak* nevezzük az olyan maximális hosszúságú (azaz se balra, se jobbra ki nem terjeszthető), legalább 3 hosszúságú számsorozatot, amelyben a szomszédos elemek értéke nem azonos. (11 pont)

```
(* dimbdombok ns = a legalább 3 hosszúságú dimbdombok száma az ns listában
   dimbdombok : int list -> int
*)
```

Példák:
```
dimbdombok [1,3,2, 2, 2,3,8, 8,8,8, 8,7,6,7, 7] = 3;
dimbdombok [5,5, 5,2,3,5, 5,5,2,2,5,5] = 1;
dimbdombok [] = 0;
```