Hanák Péter, Patai Gergely {hanak,patai}@iit.bme.hu

Deklaratív programozás, 2. gyakorlat
Erlang programozás: listák használata
2010. 09. 27−29.

Írjon olyan Erlang−eljárást, amely megfelel az adott fejkommentnek. A feladat megoldásához felhasználhat korábbi sorszámú feladatokban definiált eljárásokat.

---

1. Lista hossza

```
    Egy lista elemeinek számát a lista hosszának nevezzük.

    %% @spec listaHossza(L::egeszlista()) −> H::integer().
    %% Az L egészlista hossza H.

    listaHossza([1,3,5]) == 3.
```

---

2. Számlista minden elemének növelése

```
    %% @spec listaNoveltje(L0::egeszlista()) −> L::egeszlista().
    %% Az L egészlista az L0 egészlistának olyan másolata, amelynek ugyanannyi
    %% eleme van, mint az L0−nak, de az L minden elemének értéke pontosan
    %% eggyel nagyobb, mint az L0 megfelelõ elemének az értéke.

    listaNoveltje([1,5,2]) == [2,6,3].
```

---

3. Egy lista utolsó elemének meghatározása

```
    %% @spec listaUtolsoEleme(L::egeszlista()) −> E::integer().
    %% Az L egészlista utolsó eleme E.

    listaUtolsoEleme([5,1,2,8,7]) == 7.
```

---

4. Beszúrás listába adott helyre

```
    %% @spec insert_nth(Ls::[term()], E::term(), N::integer()) −> Rs: [term()].

    %% Az Rs lista az Ls lista olyan másolata, amelybe az Ls lista N−edik és
    %% (N+1)−edik eleme közé be van szúrva az E elem (a lista számozása 1−tõl
    %% kezdõdik).

    insert_nth([1,8,3,5], 6, 2) == [1,8,6,3,5].
    insert_nth([1,3,8,5], 3, 3) == [1,3,8,3,5].
```

---

5. Beszúrás rendezett listába

```
    %% @spec insert_ord(S0s::[integer()], E::integer()) −> Ss::[integer()].

    %% Az Ss szigorúan monoton növekvõ egészlista az S0s szigorúan monoton
    %% növekvõ egészlistának az E egésszel bõvített változata, feltéve hogy
    %% E nem eleme az S0s listának; egyébként Ss == S0s.

    insert_ord([1,3,5,8], 6) == [1,3,5,6,8].
    insert_ord([1,3,5,8], 3) == [1,3,5,8].
```


---

6. Adott lista adott sorszámú eleme

```
    %% @spec nth1_1(N::integer(), Ls::[term()]) −> E::term().
    %% Az Ls lista N−edik eleme E (1−tol számozva az elemeket).

    Az eljárás nevében szereplõ elsõ 1−es arra utal, hogy 1−tol számozzuk az
    elemeket, a második 1−es pedig arra, hogy ez az elsõ változata egy
    feladatsornak.

    nth1_1(3, [a,b,c]) == c.
```

---

7. Adott lista sorszámozott elemeiből álló lista

```
    %% @spec nth1_2(Ls::term()) −> {N::integer(), E::term()}.
    %% Az Ls lista N−edik eleme E (1−tol számozva az elemeket).

    nth1_2([a,b,c]) == [{1,a},{2,b},{3,c}].
```

---

8. Lista adott hosszúságú prefixuma

```
    Egy N elemu Xs lista prefixumának nevezzük az L hosszságú Ps listát, ha a
    Ps az Xs elsõ L elemébõl áll (az Xs−beli sorrend megtartásával), ahol 0 =<
    L =< N.

    %% @spec prefix_length(L::integer(), Xs::[term()]) −> Ps::[term()].
    %% Az Xs lista L hosszúságú prefixuma a Ps lista.

    prefix_length(3, [a,b,c,d,e]) == [a,b,c].

    Megjegyzés: a prefix_length−szel azonos funkcionalitású függvények szokásos
    neve a funkcionális nyelvekben: take.
```

---

9. Lista adott helyen kezdodo szuffixuma

```
    Egy N elemu Xs lista szuffixumának nevezzük az N−B hosszúságú Ss listát, ha
    az Ss az Xs elsõ B eleme utáni elemekbõl áll (az Xs−beli sorrend
    megtartásával), ahol 0 =< B =< N.

    %% @spec suffix_before(B::integer(), Xs::[term()]) −> Ss::[term()].
    %% Az Xs lista olyan szuffixuma Ss, amely az Xs elsõ B elemét nem
    %% tartalmazza.

    suffix_before(3, [a,b,c,d,e]) == [d,e].

    Megjegyzés: a suffix_before−ral azonos funkcionalitású függvények szokásos
    neve a funkcionális nyelvekben: drop.
```

---

10. Részlista képzése

```
    %% sublist(B::integer(), L::integer(), Xs::[term()]) −> Ps::[term()].
    %% Az Xs lista olyan (folytonos) részlistája az L hosszúságú Ps lista,
    %% amely elõtt B számú elem áll Xs−ben.

    sublist(1, 3, [a,b,c,d,e]) == [b,c,d].
```

```
--------------------------------------------------------------------------
11. Lista adott hosszúságú összes részlistáját tartalmazó lista

    %% sublist(L::integer(), Xs::[term()]) ->
    %%                    [{B::integer(), Ps::[term()], A::integer()}].
    %% Az Xs lista egy olyan (folytonos) részlistája az L hosszúságú Ps lista,
    %% amely elõtt B és amely után A számú elem áll Xs−ben.

    sublist(1,[a,b,c]) == [{0,[a],2}, {1,[b],1}, {2,[c],0}].
    sublist(2,[a,b,c]) == [{0,[a,b],1}, {1,[b,c],0}].

--------------------------------------------------------------------------
12. Lista összes nemüres részlistáját tartalmazó lista

    %% sublist(Xs::[term()]) -> [{B::integer(), Ps::[term()], A::integer()}].
    %% Az Xs lista egy olyan (folytonos), nemüres részlistája a Ps lista, amely
    %% elõtt B és amely után A számú elem áll Xs−ben.

    sublist(1,[a,b]) == [{0,[a],1}, {1,[b],0}, {0,[a,b],0}].

--------------------------------------------------------------------------
13. Lista egyre rövidülõ szuffixumainak listája

    %% @spec tails(Ls::[term()]) −> Zss::[[term()]].
    %% A Zss lista az Ls listát és egyre rövidülõ szuffixumait tartalmazza.

    %% Alternatív megoldás: tails/1 foldr−rel.

    tails([1,4,2]) == [[1,4,2],[4,2],[2],[]].
    tails([a,b,c,1,2]) == [[a,b,c,d,e],[b,c,d,e],[c,d,e],[d,e],[e],[]].

--------------------------------------------------------------------------
14. Listában párosával elõforduló elemek listája

    %% @spec parban(Xs::[term()]) −> Zs::[term()].
    %% A Zs lista az Xs lista összes olyan elemét tartalmazza, amelyet
    %% vele azonos értékû elem követ.

    %% Alternatív megoldás: parban/1 listafüggvényekkel.

    parban([a,a,a,2,3,3,a,2,b,b,4,4]) == [a,a,3,b,4].

--------------------------------------------------------------------------
15. Listában párosával elõforduló részlisták listája

    %% @spec dadogo(Xs::[term()]) −> Zss::[[term()]].
    %% A Zss lista az Xs lista összes olyan nemüres (folytonos) részlistáját
    %% tartalmazza, amelyet vele azonos értékû részlista követ.

    dadogo([a,a,a,2,3,3,a,2,b,b,b,b,4,4]) == [[a],[a],[3],[b],[b,b],[b],[b]].

--------------------------------------------------------------------------
16. Lista elsõ szigorúan monoton növekvõ részlistája (futama)

    %% @spec rampa(Xs::[term()]) −> {Zs::[term()], Ms::[term()]}.
    %% A Zs lista az Xs lista elsõ monoton növekvõ futama, az Ms az Xs maradéka.

    %%%% Segédfüggvény gyûjtõargumentummal.
    %% @spec rampa(Xs::[term()], Rs::[term()]) −> {Zs::[term()],Ms::[term()]}.
    %% A Zs lista az Xs lista elsõ monoton növekvõ futama az Rs fordítottja
    %% mögé fûzve; az Ms az Xs maradéka (Zs utáni része).
```



```
    %% Alternatív megoldás: rampa/1 splitwith függvénnyel.

    %% splitwith(Pred, List) −> {takewhile(Pred, List), dropwhile(Pred, List)}.
    %%
    %% lists:splitwith(fun erlang:is_atom/1, [a,b,c,d,1,2,3,4,e,f,5,6,7]) ==
    %%    {[a,b,c,d],[1,2,3,4,e,f,5,6,7]}.

    rampa([1,2,2,3,2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]) ==
      {[1,2,2,3],[2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]}.

--------------------------------------------------------------------------
17. Lista transzformáltja

    %% @spec trans(F::fun([term()]) −> {term(),[term()]}, Xs::([term()]))
    %%                                                    −> Zs::([term()]).
    %% A Zs lista az Xs listából az F ismételt alkalmazásával elõálló lista,
    %% ahol F párban visszaadja a Zs következõ elemét és az Xs még nem
    %% feldolgozott részét.

    trans(fun(Xs) −> lists:split(3, Xs) end, "jaaaj!!! nem jooo!") ==
      ["jaa","aj!","!! ","nem"," jo","oo!"].

--------------------------------------------------------------------------
18. Lista szigorúan monoton növekvõ részlistáinak (futamainak) listája

    % @spec rampak(Xs::[term()]) −> Xss::[[term()]].
    % Az Xss az Xs monoton növekvõ futamainak listája.

    %% Alternatív megoldás: rampak/1 trans függvénnyel.

    rampak([1,2,2,3,2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1]) ==
      [[1,2,2,3],[2,4,5,6,6,6,7],[6,8],[2,3,3,4,5,6],[0,6],[5],[4],[3],[2],[1]].

--------------------------------------------------------------------------
19. Lista számtani sorozatot alkotó prefixuma

    %% @spec dif(Ls::[number()]) −> {Ds::[number()], Zs::[number()]}.
    %% A Ds lista az Ls lista számtani sorozatot alkotó prefixuma, a Zs az Ls
    %% maradéka (Zs utáni része).

    %%%% Segédfüggvény gyûjtõargumentummal.
    %% @spec dif(Ls::[number()],N::number(),Rs::[number()]) −>
    %%                                    {Ds::[number()], Zs::[number()]}.
    %% A Ds lista az Ls lista N különbségû számtani sorozatot alkotó prefixuma
    %% az Rs lista fordítottja mögé fûzve, a Zs az Ls maradéka (Zs utáni része).

    %% Alternatív megoldás: dif/1 feltétel helyett mintaillesztéssel.

    dif([1,2,3,4,8,16,32,33,34]) == {[1,2,3,4],[8,16,32,33,34]}.
    dif([1,2,3,4,8,16,24,32,33,34]) == {[1,2,3,4],[8,16,24,32,33,34]}.
```