Deklaratív programozás, 7. gyakorlat
Erlang programozás: fa, kivétel, lusta lista
2016.11.03.

Megjegyzés: néhány feladathoz segítséget talál a feladatsor végén.

BINÁRIS FÁK

A példasorban a fa() és egeszfa() adattípusokat a következő módon definiáljuk:

```
    % @type fa()      = level | {term(),fa(),fa()}.
    % @type egeszfa() = level | {integer(),egeszfa(),egeszfa()}.
```

Tehát egy ‘fa()’ típusú Erlang-kifejezés
- vagy egy olyan adatot tartalmazó csomópont lehet, amely további két ‘fa()’ típusú értéket tartalmaz; az első a bal részfa, a második a jobb részfa, az adatot pedig címkének nevezzük;
- vagy címke nélküli levél,

Egy ‘egeszfa()’ olyan ‘fa()’, amelynek minden címkéje egész.

A példákban felhasznált változók értéke:

```
    T1 = {4,
          {3,level,level},
          {6,
           {5,level,level},
           {7,level,level}}},
    T2 = {a,
          {b, {x,level,level}, level},
          {c,
           level,
           {d,
            {x,{e,level,level},level},
            {a, {x,level,level},{x,level,level}}}}}.
```

---

1. Egészfa minden elemének növelése

```
   %% @spec fa_noveltje(F0::egeszfa()) -> F::egeszfa().
   %% Az F fa az F0 egészfának olyan másolata, amelynek ugyanannyi levele és
   %% csomópontja van, mint az F0-nak, ám F minden címkéje pontosan eggyel
   %% nagyobb, mint az F0 megfelelő címkéje.

   fa_noveltje(T1) =:= {5,{4,level,level},{7,{6,level,level},{8,level,level}}}.
```

---

2. Bináris fa tükörképe

```
   %% @spec fa_tukorkepe(F0::fa()) -> F::fa().
   %% F az F0 fa tükörképe.

   fa_tukorkepe(T1) =:= {4,{6,{7,level,level},{5,level,level}},{3,level,level}}.
```

---

3. Bináris fa legszélső címkéjének meghatározása

```
   a) %% @spec fa_balerteke(F::fa()) -> {ok, C::term()} | error.
      %% A nemüres F fa bal oldali szélső címkéje C, amelyre és minden
      %% felmenőjére igaz, hogy bal oldali gyermek; üres fa esetén ’error’.
   b) %% @spec fa_jobberteke(F::fa()) -> {ok, C::term()} | error.
      %% A nemüres F fa jobb oldali szélső címkéje C; üres fa esetén ’error’.

   fa_balerteke(T1) =:= {ok, 3}.
   fa_balerteke(level) =:= error.
   fa_jobberteke(T1) =:= {ok, 7}.
```

---

4. Bináris fa rendezettsége

Egy bináris fa rendezett, ha inorder bejárásakor a címkéi szigorúan monoton növekednek, azaz a csomópontjai kielégítik a keresőfa-tulajdonságot: minden egyes csomópont címkéje nagyobb a bal oldali gyermekei címkéinél és kisebb a jobb oldali gyermekei címkéinél. (Tipp a végén.)

```
   %% @spec rendezett_fa(F::fa()) -> B::bool().
   %% B igaz, ha az F fa rendezett.

   rendezett_fa(T1) =:= true.
   rendezett_fa(T2) =:= false.
```

---

5. Címke előfordulása (rendezetlen) bináris fában

```
   %% @spec tartalmaz(C::term(), F::fa()) -> B::bool().
   %% B igaz, ha C az F fa valamely címkéje.

   tartalmaz(x, T1) =:= false.
   tartalmaz(x, T2) =:= true.
```

---

6. Címke összes előfordulásának száma bináris fában

```
   %% @spec elofordul(C::term(), F::fa()) -> N::integer().
   %% A C címke az F fában N-szer fordul elő.

   elofordul(x, T1) =:= 0.
   elofordul(x, T2) =:= 4.
```

---

7. Bináris fa összes címkéjének útvonala

Egy adott csomópont útvonalának nevezzük azon csomópontok címkéinek listáját, amelyeken át a fa gyökerétől az adott csomópontig el lehet jutni.

```
   %% @type ut() = [term()].
   %% @spec utak(F::fa()) -> CimkezettUtak::[{term(), ut()}].
   %% A CimkezettUtak lista az F fa minden csomópontjához egy kételemű ennest
   %% társít, amelynek első eleme a csp. címkéje, második eleme a csp.
   %% útvonala. (Tipp a végén.)

   utak(T1) =:= [{4,[]},{3,[4]},{6,[4]},{5,[4,6]},{7,[4,6]}].
   utak(T2) =:= [{a,[]},
                 {b,[a]},
                 {x,[a,b]},
                 {c,[a]},
                 {d,[a,c]},
                 {x,[a,c,d]},
                 {e,[a,c,d,x]},
                 {a,[a,c,d]},
                 {x,[a,c,d,a]},
                 {x,[a,c,d,a]}].
```

---

8. Címke összes előfordulása bináris fában útvonallal

```
   %% @spec cutak(C::term(), F::fa()) -> Utak::[ut()].
   %% Utak azon csomópontok útvonalainak listája F-ben, amelyek címkéje C.

   a) oldja meg listanézettel és az utak/1 felhasználásával,
   b) oldja meg memóriatakarékosabban úgy, hogy csak a keresett útvonalakat
   tárolja az összes útvonal helyett. (Tipp a végén.)

   cutak(x, T1) =:= [].
   cutak(x, T2) =:= [{x,[a,b]},{x,[a,c,d]},{x,[a,c,d,a]},{x,[a,c,d,a]}].
```

---

9. Címkék felsorolása hatékonyan

```
   %% @spec cimkek(F::fa()) -> L::[term()].
   %% L az F címkéinek listája inorder sorrendben.

   cimkek(T1) =:= [3,4,5,6,7].
```

KIVÉTELKEZELÉS

---

10. Kivétel dobása

```
%% @spec hanyados({tort, Sz::integer(), N::integer()}) -> H::float().
%% H = Sz / N, de kivételt dob hiba esetén:
%%    error:function_clause kivételt dob, ha Sz,N nem egészek,
%%    error:badarith kivételt dob, ha N nulla.

hanyadosteszt(Fun,Tort) -> try Fun(Tort) catch error:Kiv -> Kiv end.

[badarith, function_clause, 1/2] =:= [ hanyadosteszt(fun hanyados/1, T)
    || T <- [{tort,1,0}, {tort,0.1,1}, {tort,1,2}] ].
```

---

11. Kivétel elkapása

Az iménti hanyados/1 függvény felhasználásával, annak módosítása nélkül írjon olyan függvényt, amely 0 nevező esetén az 'inf' atomot adja vissza.

```
%% @spec hanyadosinf({tort, integer(), integer()}) -> float() | inf.

[inf, function_clause, 1/2] =:= [ hanyadosteszt(fun hanyadosinf/1,T)
    || T <- [{tort,1,0}, {tort,0.1,1}, {tort,1,2}] ].
```

LUSTA LISTA

A lusta (avagy késleltetett kiértékelésű) lista típusdefiníciója:
`@type lazy:list() = [] | [Head::any()|fun() -> Tail::lazy:list()].`
Így akár végtelen hosszú listákat is létrehozhatunk.

---

12. Végtelen számtani sorozat

```
%% @spec infseq(N::integer(),D::integer()) -> L::lazy:list().
%% L lusta lista elemei: N, N+D, N+2D, ...

1> L1='dp16a-gy7':infseq(3,2).
[3|#Fun<dp16a-gy7.0.127279525>]
2> L2=(tl(L1))().
[5|#Fun<dp16a-gy7.0.127279525>]
3> L3=(tl(L2))().
[7|#Fun<dp16a-gy7.0.127279525>]
```

---

13. Hozzáférés lusta listához

```
%% @spec nth(N::integer(), L::lazy:list()) -> E::term().
%% E az L lusta lista N. eleme.

L = infseq(3,2), [nth(N,L) || N <- [1,2,3]] =:= [3,5,7].
```

---

14. Inverz faktoriális sorozat

```
%% @spec invfacs() -> L::lazy:list().
%% L lusta lista elemei: 1/0!, 1/1!, 1/2!, 1/3!, ...

L = invfacs(), [nth(I,L) || I <- [1,2,3]] =:= [1,1,1/2].
```

Használja az infseq/2 és a tanult lazy:map/2, bevezeto:fac/1 függvényeket:

```
%% @spec map(fun(), lazy:list()) -> lazy:list().
map(_, []) ->  % végtelen lista esetén fölösleges klóz
    [];
map(F, [H|T]) ->
    [F(H)|fun() -> map(F, T()) end].

fac(0) -> 1;
fac(N) -> N * fac(N-1).
```

---

15. Kumulált részletösszegek

```
%% @spec sums(L::lazy:list()) -> S::lazy:list().
%% Ha az L lusta lista elemei a1, a2, a3, ..., akkor az S lusta lista
%% elemei rendre (a1), (a1+a2), (a1+a2+a3), ...

[nth(I, sums(infseq(1, 1))) || I <- [1,2,3,4,5] ] =:= [1,3,6,10,15].
```

---

16. Euler-féle számhoz konvergálás

Adja meg, hogy adott epszilon esetén a sums(invfacs()) sorozat hányadik eleme van már a határérték epszilon sugarú környezetében.

```
%% @spec euler(Epsilon::float()) -> {A::float(), N::integer()}.
%% A sums(invfacs()) sorozat N. eleme A, melyre először |A-e^1|<Epsilon.

1> 'dp16a-gy7':euler(0.0001).
{7,2.7182539682539684}
2> abs(element(2, 'dp16a-gy7':euler(0.0001)) - math:exp(1)).
2.7860205076724043e-5
```

SEGÍTSÉG A MEGOLDÁSHOZ

4. Bináris fa rendezettsége
   A megoldásban célszerű a 3.a) és b) feladatok megoldásait segédeljárásként felhasználni, így nem szükséges további segédeljárást definiálni.
7. Bináris fa összes címkéjének útvonala
   Javasolt segédeljárás:
   `%% @spec utak(F::fa(),Eddigi::ut())->CimkezettUtak::[{C::term(),U::ut()}].`
   %% A CimkezettUtak lista az F fa minden csomópontjához egy kételemű ennest
   %% társít, amelynek első eleme (C) a csp. címkéje, második eleme (U) az
   %% Eddigi útvonal és a csp. útvonala összefűzve.
8. Címke összes előfordulása bináris fában útvonallal
   b) A megoldás nagyon hasonló a 7. megoldáshoz, de a fa gyökerének címkéjét csak feltételesen tároljuk el.
9. Cél a lineáris időigényű algoritmus. Javasolt segédfüggvény:
   `%% @spec cimkek(F::fa(), L1::[term()]) -> L::[term()].`
   %% L az F címkéinek listája inorder sorrendben L1 elé fűzve.

----- $LastChangedDate: 2016-11-02 12:30:17 +0100 (sze, 02 nov 2016) $ ------