***7. táblás gyakorlat feladatai***

1. Egy nem üres szekvenciális inputfájlban egy bank ügyfeleinek havi kivét/betét forgalmát (tranzakcióit) tárolják. Egy tranzakció az ügyfél számlaszámából, a tranzakció dátumából, és összegéből (előjeles egész szám: negatív a kivét, pozitív a betét) áll. A tranzakciók számlaszám szerint rendezetten helyezkednek el. Gyűjtsük ki azon számlaszámokat, ahol volt 100000 Ft-ot meghaladó kivétel, és adjuk meg azt a számlaszámot is, amelynél a legkisebb volt a havi forgalom (kivétek és betétek együttes darabszáma).

*Specifikáció*:

*A* = ( x:infile(Tranz), y:outfile(𝕊), sz:𝕊 ) Tranz = rec(száml:𝕊, dátum:𝕊, össz:ℤ)  
*Ef* = ( x = x0 ∧ x↗száml ∧ |x|>0)

(az x↗száml azt jelzi, hogy x számlaszám szerint növekedően rendezett)

*Ötlet:*

Soroljuk fel számlaszámonként a transzakciók összesített egyenlegeit, hogy ezekből kiválogathassuk a -100000-nél kisebb egyenlegűeket, valamint adjuk meg, hány tranzakció történt az adott számlán.

*Új specifikáció*:

*A* = ( t:enor(Egyenleg), y:outfile(𝕊), sz:𝕊 ) Egyenleg = rec(száml:𝕊, egy:ℤ, db:ℕ)  
*Ef* = ( t = t0 ∧ |t|>0 )   
*Uf* = ( y = ⊕e∊t0 <e.száml> ∧ (min, elem) = **MIN**e∊t0 e.db ∧ sz = elem.száml )

e.egy<-100000

*Összegzés (kiválogatás)*

t:enor(E) ~ tenor(Egyenleg)

f(e) ~ <e.száml> ha e.egy < -100000

s ~ y

H, +, 0 ~ 𝕊\*, ⊕, <>*Maximum kiválasztás (minimum)*

t:enor(E) ~ tenor(Egyenleg)

f(e) ~ e.db

max ~ min

H, > ~ ℤ, <

A két tételt közös felsorolás felett végezzük. De a minimum kiválasztást itt nem lehet összegzésként felírni (az ℕ halmaznak nincs maximuma), ezért a kiválogatás első lépését a minimum kiválasztáshoz hasonlóan a ciklus előtt végezzük el. (Az elem változó használata helyett elég az sz változóval dolgozni.)

*Algoritmus*:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t.first() | | | | |
| t.current().egy < -100000 | | | | |
| y : write(t.current().száml) | | | ‒ | |
| min, sz := t.current().db, t.current().száml | | | | |
| t.next() | | | | |
| ¬t.end() | | | | |
|  | t.current().egy < -100000 | | | |
|  | y : write(t.current().száml) | ‒ | | |
|  | t.current().db < min | | | |
|  | min, sz := t.current().db, t.current().száml | | | ‒ |
|  | t.next() | | | |

*Felsoroló:*

t:enor(Egyenleg) Egyenleg = rec(száml:𝕊, egy:ℤ, db:ℕ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Egyenleg\* | first() | next() | current() : Egyenleg | end() : 𝕃 |
| x : infile(Tranz)  dx : Tranz  sx : Status  akt : Egyenleg  vége : 𝕃 | sx,dx,x:read  next() | lásd külön | **return** akt | **return** vége |

Tranz = rec(száml:𝕊, dátum:𝕊, össz:ℤ)

*next() művelet*

*A* = (x:infile(Tranz), dx:Tranz, sx:Status, akt:Egyenleg, vége:𝕃 )

*Ef* = (x = x’ ∧ x↗száml ∧ dx = dx’ ∧ sx = sx’ )

*Uf* = ( vége=(sx’=abnorm) ∧ (¬vége → akt.száml=dx’.száml∧

∧ ((akt.egy, akt.db), (sx,dx,x)) ) )

*Dupla összegzés*

t:enor(E) ~ x:infile(Tranz) )  
 (sx,dx,x:read) first() nélkül,

amíg: dx.száml=akt.száml

f(e) ~ (dx.össz, 1)

s ~ (akt.egy, akt.db)

H, +, 0 ~ ( ℤ×ℕ), (+, +), (0, 0 )

|  |  |  |
| --- | --- | --- |
| vége := sx=abnorm | | |
| ¬vége | | |
| akt := (dx.száml, 0, 0) | | ‒ |
| sx=norm ∧ dx.száml=akt.száml | |
|  | akt.egy := akt.egy + dx.össz  akt.db := akt.db + 1 |
|  | sx,dx,x:read |

1. Egy szekvenciális inputfájl vadászok eredményeit (vadász neve, vadászat dátuma, zsákmány fajtája, zsákmány súlya) tárolja a vadászok neve, azon belül a vadászat dátuma szerint rendezetten.
2. Igaz-e, hogy minden vadász valamelyik vadászatán lőtt medvét?

*Specifikáció*:

*A* = ( x:infile(Trófea), l:𝕃 )

Trófea = rec(név:𝕊, dátum:𝕊, fajta:𝕊, súly:ℕ)

*Ef* = ( x=x0 ∧ x↗(név,dátum) )

*Ötlet:*

Soroljunk fel annyi logikai értéket ahány vadász van: egy logikai érték akkor igaz, ha a vadász lőtt medvét.

*Új specifikáció*:

*A* = ( t:enor(𝕃), l:𝕃)

*Ef* = ( t=t0 )

*Uf* = ( l = ∀SEARCHe∊t0 e )

*Optimista lineáris keresés*

t:enor(E) ~ t:enor(𝕃)

felt(e) ~ e

*Algoritmus*:

|  |  |
| --- | --- |
| l := igaz  t.first() | |
| l ∧ ¬t.end() | |
|  | l := t.current() |
|  | t.next() |

*Felsoroló:* t:enor(𝕃)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 𝕃\* | first() | next() | current() : 𝕃 | end() : 𝕃 |
| x : infile(Trófea)  dx : Trófea  sx : Status  akt : 𝕃  vége : 𝕃 | sx,dx,x:read next() | lásd külön | **return** akt | **return** vége |

Trófea = rec(név:𝕊, dátum:𝕊, fajta:𝕊, súly:ℕ)

*next() művelet* (Lőtt-e medvét a soron következő vadász?)

*A* = (x:infile(Trófea), dx:Trófea, sx:Status, akt:𝕃, vége:𝕃 )

*Ef* = (x = x’ ∧ x↗név ∧ dx = dx’ ∧ sx = sx’)

*Uf* = ( vége = (sx’=abnorm) ∧ (¬vége → (akt, (sx,dx,x)) dx.fajta=”medve”) )

Az eldöntéshez a lineáris keresés nem elég, mert végig kell menni egy vadász összes trófeáján.

*Összegzés (összevagyolás)*

t:enor(E) ~ x:infile(Trófea) (sx,dx,x:read)  
 first() nélkül,

amíg: dx.név=dx’.név

f(e) ~ dx.fajta=”medve”

s ~ akt

H, +, 0 ~ 𝕃 , ∨, hamis

|  |  |  |
| --- | --- | --- |
| vége := sx=abnorm | | |
| ¬vége | | |
| akt := hamis | |  |
| név := dx.név | |  |
| sx=norm ∧ dx.név=név | | ‒ |
|  | akt := akt ∨ (dx.fajta=”medve”) |  |
|  | sx,dx,x:read |  |

1. A fenti mellett mondjuk meg azt is, hogy hány olyan vadász van, hogy zsákmányai összsúlya meghaladja a 250 kilogrammot.

*Specifikáció*:

*A* = ( x:infile(Trófea), l: 𝕃, c:ℕ )

Trófea = rec(név:𝕊, dátum:𝕊, fajta:𝕊, súly:ℕ)  
*Ef* = ( x=x0 ∧ x↗(név,dátum))

*Ötlet:*

Soroljuk fel, hogy az egyes vadászok lőttek-e medvét, és mennyi a zsákmányaiknak az összsúlya.

*Új specifikáció*:

*A* = ( t:enor(Vadász), l: 𝕃, c:ℕ )

Vadász = rec(medve:𝕃, össz:ℕ)

*Ef* = ( t=t0 )

*Uf* = ( l = e∊t0e.medve ∧ c = **∑** e∊t01 )

e.össz>250

*Összeéselés és számlálás közös felsorolón*

t:enor(E) ~ t:enor(Vadász)

f(e) ~ e.medve

H, +, 0 ~ 𝕃, ∧, igaz

felt(e) ~ e.össz>250

*Algoritmus*:

|  |  |  |
| --- | --- | --- |
| l, c := igaz, 0  t.first() | | |
| ¬t.end() | | |
|  | l := l ∧ t.current().medve | |
|  | t.current().össz > 250 | |
|  | c := c + 1 | ‒ |
|  | t.next() | |

*Felsoroló*:

t:enor(Vadász) Vadász = rec(medve:𝕃, össz:ℕ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vadász\* | first() | next() | current() : Vadász | end() : 𝕃 |
| x : infile(Trófea)  dx : Trófea  sx : Status  akt : Vadász  vége : 𝕃 | sx,dx,x:read next() | lásd külön | **return** akt | **return** vége |

Trófea = rec(név:𝕊, dátum:𝕊, fajta:𝕊, súly:ℕ)

*next() művelet*

*A* = ( x : infile(Trófea), akt:Vadász, vége:𝕃 )

*Ef* = (x = x’ ∧ dx = dx’ ∧ sx = sx’ ∧ x↗(név,dátum))

*Uf* = ( vége = (sx=abnorm) ∧ ( ¬vége →

( akt.medve, (sx,dx,x)) dx.fajta=”medve” )

∧ ( akt.össz, (sx,dx,x)) ) )

*Két összegzés közös felsorolóval*

t:enor(E) ~ x:infile(Trófea)  
 (sx,dx,x:read) first() nélkül,

amíg: dx.név=dx’.név

f(e) ~ (dx.fajta=”medve”, dx.súly)

s ~ akt

H, +, 0 ~ (𝕃×ℕ), (∨, +), (hamis, 0)

|  |  |  |
| --- | --- | --- |
| vége := sx=abnorm | | |
| ¬vége | | |
| név, akt := dx.név, ( hamis, 0 ) | |  |
| sx=norm ∧ dx.név=név | | ‒ |
|  | akt.medve := akt.medve ∨  dx.fajta=”medve” |  |
|  | akt.össz := akt.össz + dx.súly |  |
|  | sx, dx, x:read |  |