Zadavanje listi

Tehnika zadavanja listi (engl. list comprehension) se koristi za rješavanje problema u funkcionalnom programiranju. Kao što samo ime kaže, zadavnjem (određivanjem) liste rješava se dati problem.

Podsjetimo se prvo matematičke notacije zadavanja skupa. Tako je npr. sa

$$\{x^2 |\ x\ \in \{1,2,3,4,5\}\}$$

zadat skup $\{1,4,9,16,25\}$.

Slično, u Haskell-u se na sljedeći način može zadati lista koja sadži iste brojeve kao i navedeni skup

$$[x\hat{}2 \mid x \leftarrow [1..5]],$$

gdje je [1..5] lista cijelih brojeva od 1 do 5, a izraz $x \leftarrow [1..5]$ nazivamo generatorom liste. Generatorom liste se definiše redoslijed „uzimanja“ elemenata iz liste, a svaki od elemenata se dalje obrađuje na način koji je određen. Tako u ovom primjeru, se prvo uzima vrijednost $x = 1$ i u rezultujuću listu se dodaje kvadrat broja 1, zatim se uzima vrijednost $x = 2$ pa se u rezultujuću listu se dodaje broj 4, itd.

Lista se može zadati i sa više generatora. Na primjer, sa

$$[(x, y) |\ x \leftarrow [1,2,3], y \leftarrow [4,5]] \qquad (*)$$

se zadaje lista $[(1,4), (1,5), (2,4), (2,5), (3,4), (3,5)]$. Dakle, uzima se prva vrijednost prvog generator (u ovom slučaju $x = 1$) i formiraju se svi mogući parovi mijenjanjem vrijednosti drugog argumenta (u ovom slučaju y), dalje se uzima druga vrijednost prvog generatora ($x = 2$) i prave se svi mogući parovi mijenjanjem vrijednosti drugog generatora.

Promjena redoslijeda generatora utiče na rezultujuću listu. Na primjer, ako zamjenimo redoslijed generatora u prethodnom primjeru, pa listu zadamo sa

$$[(x, y) |\ y \leftarrow [4,5], x \leftarrow [1,2,3]] \qquad (**)$$

onda je rezultujuća lista

$$[(1,4), (2,4), (3,4), (1,5), (2,5), (3,5)].$$

Može se primjetiti da obe liste (zadate sa (*) i (**)) sadrže iste elemente, ali ne u istom redoslijedu. Razlog za to je što se u zadavanju (**) prvo fiksira vrijednost drugog generatora y = 4, a zatim se mijenjaju vrijednosti prvog generatora $x = 1,2,3$. Nakon toga, fiksira se druga moguća vrijednost drugog generatora $y = 5$, te se ponovo formiraju parovi tako što se mijenja vrijednost prvog generatora.

Kasniji generatori mogu da zavise od promjenljivih koje su uvedene prethodnim generatorima. Na primjer, sa

$$[(x, y)|x \leftarrow [1..3], y \leftarrow [x..3]]$$

se zadaje lista

$$[(1,1), (1,2), (1,3), (2,2), (2,3), (3,3)],$$

odnosno lista svih parova (x, y) takvih da je su $x, y \in [1,2,3]$ i $y \geq x$. Generator od kojeg zavisi drugi generator, mora biti prethodno uveden.

Mogu se navesti i dodatni uslovi pri zadavanju listi. Uslovi se međusobno odvajaju znakom zapeta. Lista parnih brojeva od 1 do 100 se može zadati sa

$$[x| \ x \leftarrow [1..100], \ even \ x],$$

gdje ugrađena funkcija $even :: Int \rightarrow Bool$ ispituje da li je dati broj paran, tj. vraća *True* ako jeste, u suprotnom *False*. Slično, postoji i funkcija $odd :: Int \rightarrow Bool$ koja ispituje da li je broj koji je argument funkcije neparan.

Zadaci:

1. ***Tehnikom zadavanja listi napisati funkciju koja za argument uzima cijeli broj, a kao rezultat vraća listu djelilaca datog broja.***

Rješenje:

$djelioci :: Int \rightarrow [Int]$

$djelioci \ n = [x \mid x \leftarrow [1..n], \quad n \ `mod` \ x == 0]$

Sa x je označen potencijalni djelitelj broja n. Djelioci broja n su brojevi iz intervala od 1 do n, zato se i postavlja generator $x \leftarrow [1..n]$, uz dodatni uslov ($n \ `mod` \ x == 0$) da x dijeli n.

2. ***Definisati ugrađenu funkciju $replicate$ tehnikom zadavanja listi.***

Rješenje:

Podsjetimo se da funkcija $replicate$ ima dva argumenta cijeli broj k i proizvoljan element tipa a – x, a kao rezultat vraća listu od k elemenata koji su jednaki x.

$mojReplicate :: Int \rightarrow a \rightarrow [a]$

$mojReplicate \ k \ x = [x \mid y \leftarrow [1..k]]$

Dakle, za svaku moguću vrijednost y (a moguće vrijednosti su od 1 do k) se u rezultujuću listu dodaje po jedan element jednak x.

3. ***Definisati ugrađenu funkciju $concat$ tehnikom zadavanja listi.***

Rješenje:

Podsjetimo se da funkcija $concat$ spaja listu listi u jednu listu (npr. $concat\ [[1,2],[3],[4,5]] = [1,2,3,4,5]$).

$mojConcat :: [[a]] \rightarrow [a]$

$mojConcat\ \ xss = [x \mid xs \leftarrow xss, x \leftarrow xs]$

Prvim generatorom $xs \leftarrow xss$ se uzima lista xs iz liste listi xss, a drugim generatorom $x \leftarrow xs$ se uzima pojedinačan element x iz liste xs i taj element x se dodaje u rezultujuću listu.

4. ***Trojka (x, y, z) se naziva Pitagorejska ako važi $x^2 + y^2 = z^2$. Korištenjem tehnike zadavanja listi, definisati funkciju $pyths :: Int \rightarrow [(Int, Int, Int)]$ koja vraća listu svih pitagorejskih trojki do date granice. Na primjer, $pyths\ 10 = [(3,4,5),(4,3,5),(6,8,10),(8,6,10)]$.***

Rješenje:

$pyths :: Int \rightarrow [(Int, Int, Int)]$
$pyths\ n = [(x,y,z) | x \leftarrow [1..n], y \leftarrow [1..n], z \leftarrow [1..n], x\hat{}2 + y\hat{}2 == z\hat{}2].$

Šta se dešava ako drugi generator zamjenimo sa $y \leftarrow [x..n]$?

5. ***Pozitivan cijeli broj je savršen ako je jednak sumi svojih faktora (ne uključujući sam taj broj). Korištenjem tehnike zadavanja listi definisati funkciju koja vraća listu svih savršenih brojeva do date granice. Na primjer, $savrseni\ 500 = [6, 28, 496]$.***

Rješenje:

S obzirom da nam trebaju svi djelioci datog broja možemo koristiti funkciju iz prvog zadatka.

$savrseni :: Int \rightarrow [Int]$
$savrseni\ n = [x \mid x \leftarrow [1..n], \quad sum(init\ (djelioci\ x)) == x]$

Dakle, razmatraju se svi brojevi od 1 do n (to se postiže generatorom $x \leftarrow [1..n]$), pa se za svaki broj prvo odredi lista njegovih djelilaca ($djelioci\ x$), zatim se ugrađenom funkcijom $init$ uzmaju svi elementi te liste osim posljednjeg (osim samog x) i ugrađenom funkcijom sum se izračuna suma elemenata te liste. Na kraju se dobijena suma se uporedi sa x i ako vrijedi tražena jednakost broj x se dodaje u rezultujuću listu.

Zadaci za vježbu:

1. Napisati rekurzivnu definiciju funkcije koja vraća listu savršenih brojeva do date granice. Dozvoljeno je pisanje pomoćnih funkcija, ali i pomoćne funkcije moraju biti rekurzivne. Nije dozvoljena upotreba ugrađenih funkcija i tehnike zadavanja listi.
2. Koristeći tehniku zadavanja listi i ugrađene funkcije napisati funkciju koja računa dužinu liste.
3. Koristeći tehniku zadavanja listi napisati funkciju parni a b koja generiše listu parnih cijelih brojeva iz segmenta [a, b] i funkciju neparni a b koja generiše listu neparnih cijelih brojeva iz segmenta [a, b] .
4. Koristeći tehniku zadavanja listi napisati funkciju parovi a b c d koja generiše listu parova cijelih brojeva (x, y), takvih da x pripada segmentu [a, b], a y segmentu [c, d].
5. Koristeći tehniku zadavanja listi napisati funkciju zavisnoY a b koja generiše listu parova cijelih brojeva (x, y), takvih da x pripada segmentu [a, b], a y pripada segmentu [x, b].
6. Koristeći tehniku zadavanja listi napisati funkciju zbirPar n koja pravi listu parova (a, b) takvih da su a i b prirodni brojevi čiji je zbir jednak n. n je argument funkcije.
7. Napisati funkciju koja iz liste izdvaja listu prostih brojeva:
    a. Rekurzivno;
    b. Tehnikom zadavanja listi i ugrađenim funkcijama.

   Dozvoljeno je korištenje pomoćnih funkcija, ali ne i „preplitanje tehnika“.

8. Napisati funkciju koja ispituje da li su svi elementi liste neparni brojevi. Na primjer za [11,15,5,7] funkcija vraća True, a za [11,15,52,7] vraća False. Funkciju napisati na dva načina:
    a. Rekurzivno;
    b. Tehnikom zadavanja listi i ugrađenim funkcijama.

   Dozvoljeno je korištenje pomoćnih funkcija, ali ne i „preplitanje tehnika“.
9. Napisati funkciju koja ispituje da li je bar jedan element liste paran broj. Na primjer za [11,15,5,7] funkcija vraća False, a za [11,15,52,7] vraća True. Funkciju napisati na dva načina:
    a. Rekurzivno;
    b. Tehnikom zadavanja listi i ugrađenim funkcijama.

   Dozvoljeno je korištenje pomoćnih funkcija, ali ne i „preplitanje tehnika“.
10. Napisati funkciju koja računa sumu kvadrata parnih brojeva date liste. Funkciju napisati na dva načina:
    a. Rekurzivno;
    b. Tehnikom zadavanja listi i ugrađenim funkcijama.

   Dozvoljeno je korištenje pomoćnih funkcija, ali ne i „preplitanje tehnika“.