Lengoaiak, Konputazioa eta Sistema Adimendunak

7. gaia: Haskell – 1,6 puntu – Bilboko IITUE

2013-10-30

1 Murgilketa (0,300 puntu)

Osoak diren x eta y bi zenbaki emanda, zatitzaileak bilatzen 2tik hasita, bi zenbaki horietatik zatitzaile txikiena duena itzuliko duen zt funtzioa definitu behar da. x eta y zenbakiek 2tik hasita zatitzaile txikiena bezala zenbaki bera dutenean, funtzioak -1 balioa itzuli beharko du. x edo y 1 edo txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

$$zt :: Int \text{ -> } Int \text{ -> } Int$$
$$zt\ \ x\ \ y\ \ \ldots$$

Adibideak:

$zt\ \ 49\ \ 55\ \ =\ \ 55$	2tik hasita, 49ren zatitzaile txikiena 7 delako eta 55ena 5.
$zt\ \ 16\ \ 15\ \ =\ \ 16$	2tik hasita, 16ren zatitzaile txikiena 2 delako eta 15ena 3.
$zt\ \ 35\ \ 15\ \ =\ \ 15$	2tik hasita, 35en zatitzaile txikiena 5 delako eta 15ena 3.
$zt\ \ 16\ \ 8\ \ =$ -1	2tik hasita, 16ren eta 8ren zatitzaile txikiena 2 delako.
$zt\ \ 55\ \ 25\ \ =$ -1	2tik hasita, 55en eta 25en zatitzaile txikiena 5 delako.

Murgilketaren teknika jarraituz, osoak diren x, y eta bm hiru zenbaki emanda, zatitzaileak bilatzen bm-tik hasita, x eta y zenbakietatik zatitzaile txikiena duena itzuliko duen zt_lag funtzioa definitu behar da. x eta y zenbakiek bm-tik hasita zatitzaile txikiena bezala zenbaki bera dutenean, funtzioak -1 balioa itzuli beharko du. x edo y 1 edo txikiagoa baldin bada, errore-mezua aurkeztu beharko da. Bestalde, bm balioa 2 baino txikiagoa edo x baino handiagoa edo y baino handiagoa baldin bada ere, errore-mezua aurkeztu beharko da.

$$zt\_lag :: Int \text{ -> } Int \text{ -> } Int \text{ -> } Int$$
$$zt\_lag\ \ x\ \ y\ \ bm\ \ \ldots$$

Adibideak:

$zt\_lag\ \ 49\ \ 55\ \ 3 =\ \ 55$	3tik hasita, 49ren zatitzaile txikiena 7 delako eta 55ena 5.
$zt\_lag\ \ 49\ \ 55\ \ 6 =\ \ 49$	6tik hasita, 49ren zatitzaile txikiena 7 delako eta 55ena 11.
$zt\_lag\ \ 35\ \ 15\ \ 3\ \ =\ \ 15$	3tik hasita, 35en zatitzaile txikiena 5 delako eta 15ena 3.
$zt\_lag\ \ 35\ \ 15\ \ 4\ \ =$ -1	4tik hasita, 35en zatitzaile txikiena eta 15en zatitzaile txikiena 5 delako.

zt_lag funtzioa zt funtzioa baino orokorragoa da.

zt_lag funtzioa erabiliz zt funtzioa definitzerakoan, bm parametroa zeharkatu beharreko tartearen beheko muga bezala erabiliko da.

2 Bukaerako errekurtsibitatea (0,300 puntu)

Har dezagun honako funtzio hau:

```
ss :: [Integer] -> [Integer]
ss  []  =  []
ss  (x : s)
  | null  s   = [x]
  | otherwise  = m : (ss ((takeWhile (/= m)(x : s)) ++(tail(dropWhile (/= m)(x : s)))))
                    where  m = minimum (x : s)
```

ss funtzioak hautaketaren bidezko metodoa edo "selection sort" metodoa jarraituz, zenbaki osozko zerrenda bat ordenatzen du.

$$ss\ [5,1,8,6]\ =\ [1,5,6,8]$$

ss funtzioak ez du bukaerako errekurtsibitaterik. Bukaerako errekurtsibitatea edukitzeko, honako bi funtzio hauek definitu behar dira:

- ss funtzioak jasotzen duen parametroaz gain, emaitza bezala eraikiz joango den zerrenda gordez joateko erabiliko den bigarren parametroa duen ss_lag funtzioa. Beraz, ss_lag funtzioak, alde batetik bigarren parametroa eta bestetik, lehenengo parametro bezala emandako zerrenda ordenatuz eraikitzen den zerrenda elkartuz lortzen den zerrenda itzuli behar du.

$$ss\_lag\ [5,1,8,6]\ [7,9,2]\ =\ [7,9,2,1,5,6,8]$$

- ss_lag funtzioari egokiak diren parametroekin deituz ss funtzioak egiten duen gauza bera egingo duen ss_be funtzioa.

$$ss\_be\ [5,1,8,6]\ =\ [1,5,6,8]$$

Beraz, ss funtzioak egiten duena ss_be eta ss_lag funtzioak erabiliz egin ahal izango da.

3 Zerrenda-eraketa (1,000 puntu)

3.1. (0,150 puntu) Zenbaki arrunt denez (hau da, negatiboak ez diren zenbaki oso denez) osatutako zerrendaren hasierako azpizerrenda guztiez, hau da, 0tik abiatuz eraiki daitezkeen azpizerrenda guztiez eratutako zerrenda infinitua aurkeztuz joango den has_azpi funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} has\_azpi :: [[Integer]] \\ has\_azpi \quad \ldots \end{array}$$

Adibidea:

$$has\_azpi \quad = \quad [[0], [0,1], [0,1,2], [0,1,2,3], [0,1,2,3,4], \ldots]$$

3.2. (0,100 puntu) Zenbaki arrunt denez (hau da, negatiboak ez diren zenbaki oso denez) osatutako zerrendaren hasierako azpizerrenda guztietako elementuen baturez, hau da, 0tik abiatuz eraiki daitezkeen azpizerrenda guztietako elementuen baturez eratutako zerrenda infinitua aurkeztuz joango den has_batu funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} has\_batu :: [Integer] \\ has\_batu \quad \ldots \end{array}$$

Adibidea:

$$has\_batu \quad = \quad [0, 1, 3, 6, 10, \ldots]$$

Aukera bat aurreko ariketako has_azpi funtzioa eta aurredefinitutako sum funtzioa erabiltzea da.

3.3. (0,100 puntu) Lehenengo osagai bezala osoa eta zero edo positiboa den zenbaki bat (zenbakien ohiko ordena jarraituz) eta bigarren osagai bezala 0tik lehenengo osagaira arteko zenbakien baturaz osatutako bikoteez eratutako zerrenda infinitua kalkulatuz joango den $zenb\_batu$ funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} zenb\_batu :: [(Integer, Integer)] \\ zenb\_batu \quad \ldots \end{array}$$

Adibidea:

$$zenb\_batu \quad = \quad [(0,0), (1,1), (2,3), (3,6), (4,10), \ldots]$$

Aukera bat aurreko ariketako has_batu funtzioa eta aurredefinitutako zip funtzioa erabiltzea da.

3.4. (0,175 puntu) Osoa den x zenbaki bat emanda, x zenbakia $zenb\_batu$ zerrendan zenbatgarren bikoteko bigarrren osagaia den kalkulatzen duen $zenbatgarrena$ funtzioa definitu Haskell lengoaia erabiliz. x negatiboa baldin bada, errore-mezua aurkeztu beharko da. x negatiboa ez bada, baina hala ere $zenb\_batu$ zerrendan bigarren osagai bezala ez bada inoiz agertzen, -1 itzuli beharko da.

$$zenbatgarrena :: Integer \text{ -> } Integer$$
$$zenbatgarrena\ \ x\ \ \ldots$$

Adibideak:

$$zenbatgarrena\ \ 10\ \ =\ \ 4 \qquad\qquad zenbatgarrena\ \ 9\ \ =\ \ \text{-1}$$

Aukera bat aurreko ariketetako has_batu eta $zenb\_batu$ funtzioak eta aurredefinitutako '$elem$' eta $genericTake$ funtzioak erabiltzea da.
Beste aukera bat 3.2 ariketetako has_batu funtzioa eta aurredefinitutako $takeWhile$, '$elem$' eta $length$ funtzioak erabiltzea da.

3.5. (0,100 puntu) Zenbaki osozko bi zerrenda emanda, bigarren zerrendan ere agertzen diren lehenengo zerrendako elementuez osatutako zerrenda kalkulatzen duen $badaudenak$ funtzioa definitu Haskell lengoaia erabiliz. Agerpen-kopuruak ez dauka berdina izan beharrik lehenengo eta bigarren zerrendetan, nahikoa da agertzearekin.

$$badaudenak :: [Int] \text{ -> } [Int] \text{ -> } [Int]$$
$$badaudenak\ \ s\ \ r\ \ \ldots$$

Adibideak:

$$badaudenak\ \ [8,5,8,4,9,0]\ \ [9,7,2,8]\ \ =\ \ [8,8,9]$$
$$badaudenak\ \ [8,5,8,4,9,0]\ \ [3,6]\ \ =\ \ [\,]$$

Aukera bat aurredefinitutako '$elem$' funtzioa erabiltzea da.

3.6. (0,100 puntu) Zenbaki osozko zerrendez osatutako zerrenda bat eta zenbaki osozko zerrenda bat emanda, lehenengo parametroko zerrenda bakoitzarentzat bigarren parametroan ere badauden elementuez osatutako zerrenda kalkulatuko duen $badaude\_zerrenda$ funtzioa definitu Haskell lengoaia erabiliz.

$$badaude\_zerrenda :: [[Int]] \text{ -> } [Int] \text{ -> } [[Int]]$$
$$badaude\_zerrenda\ \ q\ \ r\ \ \ldots$$

Adibidea:

$$badaude\_zerrenda\ \ \underbrace{[[8,5,8,4,9,0],[6,20],[\,],[9,15]]}_{\text{1. parametroa}}\ \ \underbrace{[9,7,2,8]}_{\text{2. parametroa}}\ \ =\ \ [[8,8,9],[\,],[\,],[9]]$$

Aukera bat aurreko ariketako $badaudenak$ funtzioa erabiltzea da.

3.7. (0,100 puntu) Zenbaki osozko zerrendez osatutako zerrenda bat emanda, parametro horretako zerrenda bakoitzarentzat zerrenda elementu bakarrekoa al den erabakitzen duen $elem\_bakarrekoak$ funtzioa definitu Haskell lengoaia erabiliz.

$$elem\_bakarrekoak :: [[Int]] \text{ -> } [Bool]$$
$$elem\_bakarrekoak\ \ q\ \ \ldots$$

Adibideak:

$$elem\_bakarrekoak\ \ [[8,5,8,4,9,0],[6,20],[],[9,15]]\ \ =\ \ [False, False, False, False]$$
$$elem\_bakarrekoak\ \ [[8,5,8],[6],[0],[7,15],[2]]\ \ =\ \ [False, True, True, False, True]$$

Aukera bat aurredefinitutako $length$ funtzioa erabiltzea da.

3.8. (0,175 puntu) Zenbaki osozko zerrendez osatutako zerrenda bat eta zenbaki osozko zerrenda bat emanda, lehenengo parametroko zerrenda bakoitzarentzat bigarren parametroan ere agertzen den elementu bakarra al dagoen erabakitzen duen $bakoitzean\_bat$ funtzioa definitu Haskell lengoaia erabiliz.

$$bakoitzean\_bat :: [[Int]] \text{ -> } [Int] \text{ -> } Bool$$
$$bakoitzean\_bat\ \ q\ \ r\ \ \ldots$$

Adibideak:

$$bakoitzean\_bat\ \ [[8,5,8,4,9,0],[6,20],[],[9,15]]\ \ \ [9,7,2,8]\ \ =\ \ False$$
$$bakoitzean\_bat\ \ [[8,5,8,4,9,0],[6,20],[7,12,14],[9,15]]\ \ \ [15,7,20,8,30]\ \ =\ \ False$$
$$bakoitzean\_bat\ \ [[5,8,4,9,0],[6,20],[7,12,14],[9,15]]\ \ \ [15,7,20,8,30]\ \ =\ \ True$$

Aukera bat aurreko ariketetako $badaude\_zerrenda$ eta $elem\_bakarrekoak$ funtzioak eta aurredefinitutako and funtzioa erabiltzea da.