Lengoaiak, Konputazioa eta Sistema Adimendunak

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

2013-10-31

1 Murgilketa (0,300 puntu)

Murgilketaren teknika erabiliz, x eta y bi zenbaki oso eta zenbaki osoz osatutako r zerrenda emanda, x eta y zenbakietatik r zerrendan gehien agertzen dena itzultzen duen gh funtzioa definitu. Kasu berezi bezala, x eta y kopuru berean agertzen badira,-1 itzuli beharko da.

$$gh :: Int \text{ -> } Int \text{ -> } [Int] \text{ -> } Int$$
$$gh\ \ x\ \ y\ \ r\ \ \ldots$$

Adibideak:

$gh\ \ 3\ \ 5\ \ [8,5,12,7,12]\ \ =\ \ 5$
$gh\ \ 4\ \ 9\ \ [8,5,12,7,12]\ \ =\ \ -1$
$gh\ \ 12\ \ 5\ \ [8,5,12,7,12]\ \ =\ \ 12$

$gh\ \ 9\ \ 7\ \ [9,10,7,10]\ \ =\ \ -1$
$gh\ \ 9\ \ 7\ \ []\ \ =\ \ -1$

Murgilketaren teknika erabiltzeko gh_lag funtzioa definitu behar da. Funtzio honek x eta y bi zenbaki oso, zenbaki osoz osatutako r zerrenda bat eta $xkop$ eta $ykop$ beste bi zenbaki oso emanda, x zenbakiaren r zerrendako agerpen-kopurua gehi $xkop$ balioa y zenbakiaren agerpen-kopurua gehia $ykop$ baino handiagoa baldin bada, orduan x itzuliko du; x zenbakiaren r zerrendako agerpen kopurua gehi $xkop$ balioa y zenbakiaren agerpen-kopurua gehi $ykop$ baino txikiagoa baldin bada, orduan y itzuliko du; eta x zenbakiaren agerpen kopurua gehi $xkop$ eta y zenbakiaren agerpen-kopurua gehi $ykop$ berdinak badira -1 itzuliko du.

$$gh\_lag :: Int \text{ -> } Int \text{ -> } [Int] \text{ -> } Int \text{ -> } Int \text{ -> } Int$$
$$gh\_lag\ \ x\ \ y\ \ r\ \ xkop\ \ ykop\ \ \ldots$$

Ejemplos:

$gh\_lag\ \ 3\ \ 5\ \ [8,5,12,7,12]\ \ 10\ \ 2\ \ =\ \ 3$
3ren agerpen-kopurua gehi 10 (0 + 10) 5en agerpen-kopurua gehi 2 (1 + 2) baino handiagoa delako.
$gh\_lag\ \ 4\ \ 9\ \ [8,5,12,7,12]\ \ 10\ \ 10\ \ =\ \ -1$
4ren agerpen-kopurua gehi 10 (0 + 10) eta 9ren agerpen-kopurua gehi 10 (0 + 10) berdinak direlako.
$gh\_lag\ \ 4\ \ 9\ \ [8,5,12,7,12]\ \ 10\ \ 15\ \ =\ \ 9$
4ren agerpen-kopurua gehi 10 (0 + 10) 9ren agerpen-kopurua gehi 15 (0 + 15) baino txikiagoa delako.

gh_lag funtzioa gh funtzioa baino orokorragoa da.
gh_lag erabiliz $gh\ \ \ell$ definitzerakoan, $xkop$ eta $ykop$ parametroak x eta y balioen agerpenak zenbatzeko erabili behar dira hurrenez hurren.

2 Bukaerako errekurtsibitatea (0,300 puntu)

Har ditzagun honako funtzio hauek:

```
ordenatuta :: [Integer] -> Bool
ordenatuta []  =  True
ordenatuta (x : s)
  | null s   = True
  | x > (head s)   = False
  | otherwise   = ordenatuta s
```

```
burbuila :: [Integer] -> [Integer]
burbuila []  =  []
burbuila (x : s)
  | null s   = [x]
  | x ≤ (head s)   = x : (burbuila s)
  | otherwise   = (head s) : (burbuila(x : (tail s)))
```

```
bs :: [Integer] -> [Integer]
bs r
  | ordenado r   = r
  | otherwise   = bs(burbuja r)
```

ordenatuta izeneko funtzioak zerrenda bat goranzko ordenean ordenatuta al dagoen erabalitzen du. *burbuila* funtzioak zerrendako lehenengo elementua eskuinerantz mugitzen du bera baino handiagoa den balio bat aurkitu arte eta, jarraian, handiagoa den balio hori eskuinerantz mugituz doa oraindik handiagoa den beste balio bat aurkitu arte, eta abar. Azkenik, *bs* funtzioak burbuilaren metodoa edo "bubble sort" metodoa jarraituz zerrenda bat ordenatzen du, ordena gorakorrean.

Emandako definizioan, *burbuila* funtzioak ez du bukaerako errekurtsibitaterik. Honako bi funtzio hauek definitzea eskatzen da:

- *burbuila* funtzioak jasotzen duen zerrendaz gain, emaitza bezala lortuz joan beharreko zerrenda gordetzeko balioko digun bigarren zerrenda bat ere baduen *burbuila_lag* funtzioa.
- *burbuila_lag* funtzioari parametro egokiekin deituz *burbuila* funtzioak lortzen duen emaitza bera lortu beharko duen *burbuila_be* funtzioa.

Beraz, *burbuila* funtzioak egiten duena *burbuila_be* eta *burbuila_lag* funtzioen bidez egin ahal izango da.

3 Zerrenden eraketa (1,000 puntu)

Ohar garrantzitsua: Ariketa honetako ataletan, berredura definitzeko Haskell-en aurredefinituta dagoen eragilea ezingo da erabili, ezta logaritmoak kalkulatzeko gai den aurredefinitutako funtziorik ere.

3.1. (0,100 puntu) x eta n bi zenbaki oso emanda, x balioa n aldiz errepikatuz lortzen den zerrenda itzultzen duen $konst$ funtzioa definitu Haskell-ez. n-ren balioa 0 bada, zerrenda hutsa itzuli beharko da eta n-ren balioa negatiboa baldin bada, errore-mezua aurkeztu beharko da.

$$konst :: Integer \text{ -> } Integer \text{ -> } [Integer]$$
$$konst\ \ x\ \ n\ \ \ldots$$

Adibideak:

$$konst\ \ 10\ \ 4\ \ =\ \ [10, 10, 10, 10]$$
$$konst\ \ 10\ \ 0\ \ =\ \ [\,]$$

3.2. (0,150 puntu) Osoa den x zenbaki bat emanda, x errepikatuz lor daitezkeen luzera guztietako zerrendez osatutako zerrenda infinitua aurkeztuz joango den $errepikatu$ izeneko funtzioa definitu Haskell-ez.

$$errepikatu :: Integer \text{ -> } [[Integer]]$$
$$errepikatu\ \ x\ \ \ldots$$

Adibidea:

$$errepikatu\ \ 3\ \ =\ \ [[], [3], [3, 3], [3, 3, 3], [3, 3, 3, 3], \ldots]$$

Aurreko ariketan definitutako $konst$ funtzioa erabiltzea da aukera bat.

3.3. (0,150 puntu) Osoa den x zenbaki bat emanda, x zenbakiaren berredura denez osatutako zerrenda infinitua aurkeztuz joango den $berredura\_zerrenda$ izeneko funtzioa definitu Haskell-ez.

$$berredura\_zerrenda :: Integer \text{ -> } [Integer]$$
$$berredura\_zerrenda\ \ x\ \ \ldots$$

Adibidea:

$$berredura\_zerrenda\ \ 3\ \ =\ \ [1, 3, 9, 27, 81, \ldots]$$

Aurreko ariketan definitutako $errepikatu$ funtzioa eta aurredefinitutako $product$ funtzioa erabiltzea da aukera bat.

3.4. (0,100 puntu) Osoa den n zenbaki bat emanda, 2 zenbakiaren lehenengo n berredurez osatutako zerrenda itzuliko duen bi_ber_zer funtzioa definitu Haskell-ez. n-ren balioa 0 baldin bada, zerrenda hutsa itzuli beharko da eta n-ren balioa negatiboa baldin bada, errore-mezua aurkeztu beharko da.

$$bi\_ber\_zer :: Integer \text{ -> } [Integer]$$
$$bi\_ber\_zer\ \ n\ \ \ldots$$

Adibideak:

$$bi\_ber\_zer\ \ 5\ \ =\ \ [1, 2, 4, 8, 16] \qquad\qquad bi\_ber\_zer\ \ 0\ \ =\ \ []$$

Aurreko ariketan definitutako $berredura\_zerrenda$ funtzioa eta aurredefinitutako $genericTake$ funtzioa erabiltzea da aukera bat.

3.5. (0,100 puntu) Osoa den n zenbaki bat emanda, lehenengo n zenbakientzat, zenbakia eta zenbaki horri dagokion 2ren berreduraz eratutako bikoteez osatutako zerrenda itzuliko duen $bi\_ber\_indizea$ funtzioa definitu Haskell-ez. n-ren balioa 0 baldin bada, zerrenda hutsa itzuli beharko da eta n-ren balioa negatiboa baldin bada, errore-mezua aurkeztu beharko da.

$$bi\_ber\_indizea :: Integer -> [(Integer, Integer)]$$
$$bi\_ber\_indizea\ n\ = \ldots$$

Adibideak:

$bi\_ber\_indizea\ 5\ =\ [(0,1),(1,2),(2,4),(3,8),(4,16)]$ $\qquad$ $bi\_ber\_indizea$ -3 $=$ []

Aurreko ariketan definitutako bi_ber_zer funtzioa eta aurredefinitutako zip funtzioa erabiltzea da aukera bat.

3.6. (0,100 puntu) Osoa den x zenbakia emanda, x zenbakia 2ren berredura al den erabakitzen duen bi_ber_da funtzioa definitu Haskell-ez. x balioa 0 edo negatiboa baldin bada, errore-mezua aurkeztu beharko da.

$$bi\_ber\_da :: Integer \text{ -> } Bool$$
$$bi\_ber\_da\ x\ \ldots$$

Adibideak:

$$bi\_ber\_da\ 5\ =\ False \qquad\qquad bi\_ber\_da\ 8\ =\ True$$

3.4 ariketan definitutako bi_ber_zer funtzioa eta aurredefinitutako '$elem$' funtzioa erabiltzea da aukera bat.
Beste aukera bat, 3.3 ariketan definitutako $berredura\_zerrenda$ funtzioa eta aurredefinitutako '$elem$' eta $takeWhile$ funtzioak erabiltzea da.

3.7. (0,200 puntu) Osoa den x zenbakia emanda, x zenbakia lortzeko 2 zenbakiak zein berretzaile behar duen kalkulatzen duen $berretzailea$ funtzioa definitu Haskell-ez. x balioa 0 edo negatiboa baldin bada, errore-mezua aurkeztu beharko da. x balioa 1 edo handiagoa baldin bada baina 2ren berredura ez bada, beste errore-mezu desberdin bat aurkeztu beharko da.

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

Adibideak:

$$berretzailea\ 8\ =\ 3 \qquad\qquad berretzailea\ 32\ =\ 5$$

Aukera bat, aurreko ariketetan definitutako bi_ber_da eta $bi\_ber\_indizea$ funtzioak eta aurredefinitutako $head$ funtzioa erabiltzea da.
Beste aukera bat, aurreko ariketetan definitutako bi_ber_da eta $berredura\_zerrenda$ funtzioak eta aurredefinitutako $takeWhile$ eta $genericLength$ funtzioak erabiltzea da.

3.8. (0,100 puntu) Zenbaki osozko zerrenda bat emanda, zenbaki horietako bakoitza 2ren berredura al den ala ez erabaki eta erabaki horiez osatutako zerrenda itzuliko duen bi_ber_dira izeneko funtzioa definitu Haskell-ez. Datu bezala emandako zerrendan negatiboa den balioren bat edo 0 zenbakia agertzen bada, errore-mezua aurkeztu beharko da.

$$bi\_ber\_dira :: [Integer] \text{ -> } [Bool]$$
$$bi\_ber\_dira\ q\ \ldots$$

Adibidea:

$$\begin{array}{rcl} bi\_ber\_dira\ [8,5,8,4,9,10] & = & [True, False, True, True, False, False] \\ bi\_ber\_dira\ [8,32,16] & = & [True, True, True] \end{array}$$

Aukera bat, 3.6 ariketan definitutako bi_ber_da funtzioa eta aurredefinitutako $length$ funtzioa erabiltzea da.