

```
module Lksa_2016_11_16 where
```

```
import Data.List
```

```
-----
--MURGILKETA
-----
```

```
{-
```

```
Osoa eta positiboa den z zenbaki bat hartuta, z-ren ondoren jarraian dauden zenbaki batzuk
(gutxienez zenbaki bat) batuz z-ren aurreko zenbaki positibo denen batura bera lortzen bada,
orduan z erdikoa dela esan ohi da.
```

```
-}
```

```
{-
```

```
Datu bezala z zenbaki oso bat emanda, z zenbakia erdikoa al den erabakiko duen erdikoa_al_da
funtzioa definitu nahi da. Emandako z zenbakia positiboa ez bada, errore-mezua aurkeztu beharko da.
```

```
-}
```

```
{-
```

```
Murgilketaren teknika erabiliz, positiboa eta osoa den z zenbakia eta z baino handiagoa edo z-ren
berdina den w zenbaki osoa emanda,  $(z + 1) + (z + 2) + \dots + w$  balioari w-ren ondoren jarraian
dauden zenbaki batzuk (gutxienez zenbaki bat) batuz z-ren aurreko zenbaki positibo denen batura
bera lor al daitekeen erabakitzen duen erdikoa_al_da_lag funtzioa definitu behar da. Emandako
z zenbakia positiboa ez bada edo emandako w zenbakia z baino txikiagoa baldin bada, errore-mezua
aurkeztu beharko da. Kontuan hartu  $w = z$  betetzen baldin bada,  $(z + 1) + (z + 2) + \dots + w = 0$ 
izango dela.
```

```
-}
```

```
{-
```

```
erdikoa_al_da_lag funtzioa erdikoa_al_da baino orokorragoa da, w parametroaren bidez z-ren
ondoren jarraian dauden zenbakietatik gutxienez zenbat batu nahi diren finkatzeko aukera
ematen baitu: gutxienez  $(z + 1) + (z + 2) + \dots + w$  batura hartu beharko da abiapuntu bezala,
hau da, gutxienez,  $(z + 1)$ ,  $(z + 2)$ ,  $\dots$ , w zenbakiak batu behar dira.
```

```
-}
```

```
erdikoa_al_da_lag :: Integer -> Integer -> Bool
```

```
erdikoa_al_da_lag z w
```

```
    | z <= 0           = error "Lehenengo datua ez da positiboa."
    | w < z            = error "Bigarren datua lehenengoa baino txikiagoa da."
    | (sum [1..(z-1)]) == ((sum [(z+1)..w]) + (w+1))    = True
    | (sum [1..(z-1)]) < ((sum [(z+1)..w]) + (w+1))    = False
    | otherwise        = erdikoa_al_da_lag z (w+1)
```

```
-----

erdikoa_al_da :: Integer -> Bool
```

```
erdikoa_al_da z = erdikoa_al_da_lag z z
```

```
-----
```

```
-----
--BUKAERAKO ERREKURTSIBITATEA
-----
```

```
zip_berria :: Integer -> [Integer] -> [Integer] -> [(Integer, Integer)]
zip_berria z r s
  | (z < 0) || (z > (minimum [(genericLength r), (genericLength s)])) = error "1. datua ez da egokia."
  | (z == 0) = [ ]
  | otherwise = (((head r), (head s)) : (zip_berria (z - 1) (tail r) (tail s)))
```

```
{-
zip_berria funtzioak zenbaki osozko r eta s zerrendetan posizio berean dauden lehenengo
z elementuekin bikoteak eratzen ditu.
-}
```

```
{-
zip_berria funtzioak ez du bukaerako errekurtsibitaterik. Bukaerako errekurtsibitatea
edukitzeko, honako bi funtzio hauek definitu behar dira:
```

```
* zip_berria_lag funtzioa: funtzio horrek zip_berria funtzioak jasotzen dituen z
zenbakia eta r eta s zenbakizko bi zerrendetaz gain, emaitza bezala eraikiz joango
den bikote-zerrenda gordez joateko erabiliko den b bikote-zerrenda izango du
laugarren parametro bezala. Beraz, zip_berria_lag funtzioak b zerrenda eta zenbaki
osozko r eta s zerrendetan posizio berean dauden lehenengo z elementuekin
osatutako bikoteak dituen zerrenda elkartzuz lortzen den bikote-zerrenda itzuliko du.
z negatiboa baldin bada edo z-ren balioa r eta s zerrendetatik laburrena denaren
luzera baino handiagoa baldin bada, errore-mezua aurkeztu beharko du.
```

```
* zip_berria_be funtzioa: funtzio horrek zip_berria funtzioak egiten duen gauza bera
egin beharko du zip_berria_lag funtzioari egokiak diren parametroekin deituz.
```

```
-}
```

```
{-
Beraz, zip_berria funtzioak egiten duena zip_berria_be eta zip_berria_lag funtzioak
erabiliz egin ahal izango da.
-}
```

```
zip_berria_lag :: Integer -> [Integer] -> [Integer] -> [(Integer, Integer)] -> [(Integer, Integer)]
zip_berria_lag z r s b
  | (z < 0) || (z > (minimum [(genericLength r), (genericLength s)])) = error "1. datua ez da egokia."
  | (z == 0) = b
  | otherwise = zip_berria_lag (z - 1) (tail r) (tail s) (b ++ [((head r), (head s))])
```

```
zip_berria_be :: Integer -> [Integer] -> [Integer] -> [(Integer, Integer)]
zip_berria_be z r s = zip_berria_lag z r s [ ]
```

```
-----
```

```
-----
--ZERRENDA-ERAKETA
-----
```

```
--3.1
```

```
{-
z zenbaki osoa eta zenbaki osozko zerrendez eratutako s zerrenda emanda,
s-ko zerrenda bakoitzari z zenbakia ezkerretik erantsiz lortzen diren
zerrenda denez eratutako zerrenda itzuliko duen "erantsi" izeneko
funtzioaren definizioa. Hor, s zerrenda hutsa baldin bada, zerrenda
hutsa itzuli beharko da.
-}
```

```
erantsi :: Integer -> [[Integer]] -> [[Integer]]
```

```
erantsi z s = [ z:x | x <- s]
```

```
-----
--3.2
```

```
{- Zenbaki osozko r zerrenda eta zenbaki osozko zerrendez eratutako s
zerrenda emanda, s-ko zerrenda bakoitzari r-ko elementu bakoitza
ezkerretik erantsiz lortzen diren zerrenda denez eratutako zerrenda
itzuliko duen "erantsi_bakoitza" izeneko funtzioaren definizioa.
r zerrenda hutsa baldin bada, errore-mezua aurkeztu beharko da.
s zerrenda hutsa baldin bada, zerrenda hutsa itzuli beharko da.
-}
```

```
erantsi_bakoitza :: [Integer] -> [[Integer]] -> [[Integer]]
```

```
erantsi_bakoitza r s
  | null r      = error "Lehenengo zerrenda hutsa da."
  | otherwise   = [ y:x | y <- r, x <- s]
```

```
-- Beste aukera bat:
```

```
erantsi_bakoitza2 :: [Integer] -> [[Integer]] -> [[Integer]]
```

```
erantsi_bakoitza2 r s
  | null r      = error "Lehenengo zerrenda hutsa da."
  | otherwise   = concat [ erantsi y s | y <- r]
```

```
-----
--3.3
```

```
{- Zenbaki osozko r zerrenda eta zenbaki osozko zerrendez eratutako s
zerrenda emanda, s-ko zerrenda denez eta s-ko zerrenda bakoitzari r-ko
elementu bakoitza ezkerretik erantsiz lortzen diren zerrenda denez
eratutako zerrenda itzuliko duen "erantsi_mantenduz" izeneko funtzioaren
definizioa. r zerrenda hutsa baldin bada, errore-mezua aurkeztu beharko
da. s zerrenda hutsa baldin bada, zerrenda hutsa itzuli beharko da.
-}
```

```
erantsi_mantenduz :: [Integer] -> [[Integer]] -> [[Integer]]
```

```
erantsi_mantenduz r s
  | null r      = error "Lehenengo zerrenda hutsa da."
  | otherwise   = s ++ (erantsi_bakoitza r s)
```

```
-----
```

```
--3.4
```

```
{- Zenbaki osozko r zerrenda eta zenbaki osozko zerrendez eratutako s
zerrenda emanda, s-ko zerrenda denez eta s-ko zerrenda bakoitzari r-ko
elementu bakoitza ezkerretik behin eta berriz erantsiz lortzen diren
zerrenda denez eratutako zerrenda itzuliko duen "behin_eta_berriz"
izeneko funtzioaren definizioa. r zerrenda hutsa baldin bada,
errore-mezua aurkeztu beharko da. s zerrenda hutsa baldin bada,
zerrenda hutsa itzuli beharko da.
-}
```

```
{- Oro har, infinitua izango den zerrenda itzuliko du "behin_eta_berriz"
funtzioak.
-}
```

```
behin_eta_berriz :: [Integer] -> [[Integer]] -> [[Integer]]
```

```
behin_eta_berriz r s = s ++ (behin_eta_berriz r (erantsi_bakoitza r s))
```

```
-----
```

```
--3.5
```

```
{- Zenbaki osozko zerrendez eratutako s zerrenda emanda, lehenengo
osagai bezala 0 zenbakia duten zerrendak kenduz gelditzen den
zerrenda itzuliko duen "zerodunak_kendu" izeneko funtzioaren definizioa.
Salbuespena 0 bakar batez osatutako zerrenda izango da, hasu
horretan zerrenda ez baita kendu behar. s zerrendan zerrenda
hutsa agertzen baldin bada, errore-mezua aurkeztu beharko da.
s zerrenda hutsa baldin bada, zerrenda hutsa itzuli beharko da.
-}
```

```
zerodunak_kendu :: [[Integer]] -> [[Integer]]
```

```

zerodunak_kendu s
  | [] `elem` s    = error "Zerrenda horretan zerrenda hutsa agertzen da."
  | otherwise      = [x | x <- s, (((genericLength x) == 1) || ((head x) /= 0))]

```

```
-----
```

```
--3.6
```

```

{- Zenbaki osoz eratutako s zerrenda emanda, ezkerretik hasi eta lehenengo
berretzaile bezala 0 hartuz eta gero berretzaileak unitateka handituz,
s-ko elementu bakoitza dagokion 10 zenbakiaren berreturaz biderkatuz
lortzen den zerrenda itzuliko duen "berreturak" izeneko funtzioaren definizioa.
s zerrenda hutsa baldin bada, zerrenda hutsa itzuli beharko da.
-}

```

```
berreturak :: [Integer] -> [Integer]
```

```
berreturak s = [x * (10 ^ y) | (x,y) <- zip s [0..((genericLength s) - 1)]]
```

```
-----
```

```
--3.7
```

```

{- Digituz (0 eta 9ren arteko zenbaki osoz) eratutako s zerrenda emanda,
zerrendak adierazten duen zenbakizko balioa itzuliko duen "zenbakizko_balioa"
izeneko funtzioa definitu. s zerrendan digitua ez den zenbakiren bat baldin
badago, errore-mezua aurkeztu beharko da. s zerrenda hutsa baldin bada,
errore-mezua aurkeztu beharko da.
-}

```

```
zenbakizko_balioa :: [Integer] -> Integer
```

```

zenbakizko_balioa s
  | null s          = error "Zerrenda hori hutsa da."
  | (genericLength [x | x <- s, x `notElem` [0..9]]) /= 0    =
    error "Zerrenda horretan digitua ez den zenbakiren bat dago."
  | otherwise       = sum (berreturak (reverse s))

```

```
-----
```