```
module Zerrenda_eraketa2 where
import Data.List
import Zerrenda_eraketa
import Zerrendak

-------------------------------------

--1
-- Osoa den n zenbaki oso bat emanda, lehenengo n zenbaki
-- lehenez osatutako zerrenda itzultzen duen funtzioa.
-- Sarrera bezala emandako zenbakia negatiboa baldin bada,
-- errore-mezua aurkeztuko da.

lehenengo_lehenak:: Integer -> [Integer]

lehenengo_lehenak n
        | n < 0         = error "Datua negatiboa da"
        | otherwise     = genericTake n [ x | x <- [1..], lehena_ze x]

-------------------------------------

--2
-- Zenbaki lehen denez osatutako zerrenda infinitua aurkeztuz joango den
-- funtzioa.

lehen_denak:: [Integer]

lehen_denak = [ x | x <- [1..], lehena_ze x]

-------------------------------------

--3
-- Osoa den n zenbaki oso bat emanda, n baino txikiagoak diren
-- zenbaki lehenez osatutako zerrenda itzultzen duen funtzioa.
-- Sarrera bezala emandako zenbakia 0 edo negatiboa baldin bada,
-- zerrenda hutsa aurkeztuko da.

lehen_txikiagoak:: Integer -> [Integer]

lehen_txikiagoak n = [ x | x <- [1..(n - 1)], lehena_ze x]

-------------------------------------

--4
-- Zenbaki lehen bakoitzaz eta zenbaki hori zenbaki lehenen
-- zerrendan zenbatgarrena den adierazten duen zenbakiaz
-- osatutako bikoteen zerrenda infinitua aurkeztuz joango den
```

```
-- funtzioa.

lehen_zenbatuak:: [(Integer, Integer)]

lehen_zenbatuak = zip [1..] lehen_denak

-------------------------------------

--5
-- Osoa den n zenbaki bat emanda, lehenengo n zenbaki lehenentzako,
-- zenbakia bera eta zenbaki hori zenbaki lehenen zerrendan
-- zenbatgarrena den adierazten duen zenbakiaz osatutako bikoteen
-- zerrenda finitua aurkeztuko duen funtzioa. Emandako n
-- balioa negatiboa baldin bada, errore-mezua aurkeztuko da.

lehenengo_lehen_zenbatuak:: Integer -> [(Integer, Integer)]

lehenengo_lehen_zenbatuak n
        | n < 0         = error "Datua negatiboa da"
        | otherwise     = genericTake n lehen_zenbatuak

-------------------------------------

--6

--(LEHENENGO AUKERA)
-- Osoak diren n eta p zenbakiak emanda, p-garren zenbaki
-- lehenetik hasita lehenengo n zenbaki lehenez eta zenbaki
-- horietako bakoitza zenbatgarrena den adierazten duen
-- zenbakiez osatutako bikoteen zerrenda finitua aurkeztuko
-- duen funtzioa.
-- Emandako n balioa negatiboa bada edo p positiboa ez bada,
-- errore-mezua aurkeztuko da.

lehenak_nondik:: Integer -> Integer -> [(Integer, Integer)]

lehenak_nondik n p
        | n < 0         = error "Lehenengo datua negatiboa da"
        | p < 1         = error "Bigarren datua 1 baino txikiagoa da"
        | otherwise     = genericTake n [(x,y) | (x,y) <- lehen_zenbatuak, x >= p]

-------------------------------------

--(BIGARREN AUKERA)
-- Osoak diren n eta p zenbakiak emanda, p-garren zenbaki
-- lehenetik hasita lehenengo n zenbaki lehenez eta zenbaki
-- horietako bakoitza zenbatgarrena den adierazten duen
```

```
-- zenbakiez osatutako bikoteen zerrenda finitua aurkeztuko
-- duen funtzioa.
-- Emandako n balioa negatiboa bada edo p positiboa ez bada,
-- errore-mezua aurkeztuko da.

lehenak_nondik2:: Integer -> Integer -> [(Integer, Integer)]

lehenak_nondik2 n p
        | n < 0         = error "Lehenengo datua negatiboa da"
        | p < 1         = error "Bigarren datua 1 baino txikiagoa da"
        | otherwise     = genericTake n (genericDrop (p - 1) lehen_zenbatuak)

-------------------------------------

--7
-- Osoa den p zenbaki bat emanda, p-garren zenbaki lehena itzultzen
-- duen funtzioa. Emandako p balioa 0 edo negatiboa baldin bada,
-- errore-mezua aurkeztuko da.

lehena_aukeratu:: Integer -> Integer

lehena_aukeratu p
        | p < 1         = error "Datua ez da positiboa"
        | otherwise     = leh (genericDrop (p - 1) (lehenengo_lehenak p))

-------------------------------------

--8
-- Zenbaki osozko s zerrenda bat emanda, s zerrendan dauden
-- zenbakiez eta zenbaki horiei dagozkien zenbaki lehenez osatutako
-- bikoteak dituen zerrenda itzultzen duen funtzioa.
-- Bikoteak posizioaren arabera ordenatuta egon behar dute.
-- Gainera, s zerrendan balio bat errepikatuta agertzen bada ere,
-- balio horri dagokion bikotea behin bakarrik agertuko da.
-- Sarrerako zerrenda hutsa bada edo balio positiborik ez badu,
-- zerrenda hutsa itzuliko da.
-- 0 eta zenbaki negatiboak ez dira kontuan hartuko.

lehenak_aukeratu1:: [Integer] -> [(Integer, Integer)]

lehenak_aukeratu1 s
        | hutsa_da [x | x <- s, x >= 1]         = []
        | otherwise                     = [(y, z) | (y, z) <- (genericTake (maximum s) lehen_zenbatuak), y `elem`
s]

-------------------------------------
```

```
--9
-- Zenbaki osozko s zerrenda bat emanda, s zerrendan dauden zenbakiez eta
-- zenbaki horiei dagozkien zenbaki lehenez osatutako bikoteak dituen
-- zerrenda itzultzen duen funtzioa.
-- Bikoteak s zerrendako ordena jarraituz egongo dira.
-- Gainera, s zerrendan balio bat errepikatuta agertzen bada,
-- balio horri dagokion bikotea ere errepikatuta agertuko da.
-- Sarrerako zerrenda hutsa bada edo balio positiborik ez badu,
-- zerrenda hutsa itzuliko da.
-- 0 eta zenbaki negatiboak ez dira kontuan hartuko.

lehenak_aukeratu2:: [Integer] -> [(Integer, Integer)]

lehenak_aukeratu2 s = zip [x | x <- s, x >= 1] [lehena_aukeratu y | y <- s, y >= 1]

-------------------------------------

--10
-- Lehena izan beharko lukeen zenbaki oso bat emanda, zenbaki lehen horri
-- zenbaki lehenen zerrendan dagokion posizioa itzultzen duen funtzioa.
-- Sarrerako zenbakia ez bada lehena, errore-mezua aurkeztuko da.

lehenaren_posizioa:: Integer -> Integer

lehenaren_posizioa n
        | not (lehena_ze n)     = error "Ez da lehena"
        | otherwise             = genericLength (takeWhile (<= n) lehen_denak)

-------------------------------------

--11
-- Osoa den n zenbakia emanda, negatiboak ez diren eta beraien arteko
-- batura bezala n duten zenbakiz osatutako bikoteez eratutako
-- zerrenda aurkeztuko duen funtzioa.
-- Datu bezala emandako n zenbakia negatiboa baldin bada,
-- errore-mezua aurkeztuko da.

batura_bera:: Integer -> [(Integer, Integer)]

batura_bera n
        | n < 0         = error "Datua negatiboa da"
        | otherwise     = [(x,y) | x <- [0..n], y <- [0..n], x + y == n]

-------------------------------------

--12
-- Zenbaki arrunten bikoteez eratutako zerrenda infinitua orden egokian
```

```
-- aurkeztuz joango den funtzioa: [(0,0), (0,1), (1,0), (0,2), (1,1),
-- (2,0), (0,3), (1,2), (2,1), (3,0), (0,4), (1,3), (2,2), (3,1), (4,0), ...]

n_bider_n:: [(Integer, Integer)]

n_bider_n = concat [batura_bera x | x <- [0..]]

-------------------------------------

--13
-- Osoa eta >= 0 den n zenbaki bat emanda, n_bider_n izeneko
-- funtzioak sortzen duen zerrenda infinituko lehenengo n bikoteak
-- dituen zerrenda itzultzen duen funtzioa.
-- Emandako n balioa negatiboa denean errore-mezua aurkeztuko da.

bikote_kop:: Integer -> [(Integer, Integer)]

bikote_kop n
        | n < 0         = error "Datua negatiboa da"
        | otherwise     = genericTake n n_bider_n

-------------------------------------

--14
-- n_bider_n izeneko funtzioak itzultzen duen zerrendako bikote bakoitza
-- zerrendan zenbatgarrena den adieraziz osatutako bikoteez eratutako
-- zerrenda itzultzen duen funtzioa.

bikote_zenbatuak:: [(Integer, (Integer, Integer))]

bikote_zenbatuak = zip [1..] n_bider_n

-------------------------------------

--15
-- Osoa eta >= 0 den n zenbaki bat emanda, bikote_zenbatuak izeneko
-- funtzioak itzultzen duen zerrendako lehenengo n elementuez
-- osatutako zerrenda itzultzen duen funtzioa.
-- Emandako n balioa negatiboa denean errore-mezua aurkeztuko da.

zenbat_zenbatu:: Integer -> [(Integer, (Integer, Integer))]

zenbat_zenbatu n
        | n < 0         = error "Datua negatiboa da"
        | otherwise     = genericTake n bikote_zenbatuak

-------------------------------------
```

```
--16
-- Zenbaki osozko bikoteez osatutako s zerrenda bat emanda,
-- bikoteetako lehenengo osagaiez eratutako zerrenda itzuliko
-- duen funtzioa.

ezk:: [(Integer, Integer)] -> [Integer]

ezk s = [x | (x,y) <- s]

-------------------------------------

--17
-- Osoa den n zenbaki bat emanda, (0, 0)-tik (n, n)-rainoko
-- elementu berdinez osatutako bikoteez eratutako zerrenda
-- itzuliko duen funtzioa.
-- Emandako n balioa negatiboa baldin bada, zerrenda hutsa itzuliko da.

bikoiztu:: Integer -> [(Integer, Integer)]

bikoiztu n = [(x, x) | x <- [0..n]]

-------------------------------------

--18
-- Osoak diren x eta n bi zenbaki emanda, x zenbakiaren n
-- errepikapen dituen zerrenda itzultzen duen funtzioa.
-- Emandako n balioa 0 baldin bada, zerrenda hutsa itzuliko da.
-- Bestalde, n negatiboa baldin bada, errore-mezua aurkeztuko da.

konst:: Integer -> Integer -> [Integer]

konst x n
        | n < 0         = error "Bigarren datua negatiboa da"
        | otherwise     = [x | y <- [1..n]]

-------------------------------------

--19
-- Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, gutxienez
-- osagaietako bat 0 duten bikoteez eratutako zerrenda itzuliko duen
-- funtzioa.

zero:: [(Integer, Integer)] -> [(Integer, Integer)]

zero s = [(x, y) | (x, y) <- s, (x == 0 || y == 0)]
```

```
-------------------------------------

--20
-- Zenbaki osozko bikoteez osatutako s zerrenda bat emanda,
-- osagaien ordena trukatuz lortzen diren bikoteez eratutako
-- zerrenda itzuliko duen funtzioa.

trukatu:: [(Integer, Integer)] -> [(Integer, Integer)]

trukatu s = [(y, x) | (x, y) <- s]

-------------------------------------

--21
-- Zenbaki osozko s zerrenda bat emanda, s-ko elementu bikoitiak
-- errepikatuz osatutako bikoteez eratutako zerrenda
-- itzuliko duen funtzioa.

f:: [Integer] -> [(Integer, Integer)]

f s = [(x, x) | x <- s, (x `mod` 2 == 0)]

-------------------------------------

--22
-- Zenbaki osozko s zerrenda bat emanda, s-ko elementu bakoitzari 1 gehituz
-- lortzen den zerrenda itzuliko duen funtzioa.

inkr_ze:: [Integer] -> [Integer]

inkr_ze s = [x + 1 | x <- s]

-------------------------------------

--23
-- Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, bikote bakoitzeko
-- osagaiak bidertuz lortzen diren balioez eratutako zerrenda itzuliko duen
-- funtzioa.

bid_ze:: [(Integer, Integer)] -> [Integer]

bid_ze s = [x * y | (x, y) <- s]

-------------------------------------

--24
-- Zenbaki osozko bikoteez osatutako s zerrenda bat emanda,
```

```
-- bikote bakoitzeko lehenengo osagaia errepikatuz lortzen diren
-- bikoteez eratutako zerrenda itzuliko duen funtzioa.

leh_os:: [(Integer, Integer)] -> [(Integer, Integer)]

leh_os s = [(x, x) | (x, y) <- s]

-------------------------------------

--25
-- Osoa den n zenbaki bat eta zenbaki osozko s zerrenda bat emanda, n baino
-- handiagoak diren s-ko elementuez eratutako zerrenda itzultzen duen funtzioa.

handiagoak:: Integer -> [Integer] -> [Integer]

handiagoak n s = [x | x <- s, x > n]

-------------------------------------

--26
-- Zenbaki osozko s zerrenda bat eta osoa den x zenbaki bat eta emanda, x
-- balioaren desberdinak diren s-ko elementuez eratutako zerrenda
-- itzultzen duen funtzioa.

desberdinak:: [Integer] -> Integer -> [Integer]

desberdinak s x = [y | y <- s, y /= x]

-------------------------------------

--27
-- Osoa den n zenbaki bat eta zenbaki osozko bikoteez osatutako s zerrenda bat
-- emanda, lehenengo osagai bezala n eta bigarrena osagai bezala zenbaki
-- negatibo bat duten s-ko bikoteez osatutako zerrenda itzultzen duen funtzioa.

neg:: Integer -> [(Integer, Integer)] -> [(Integer, Integer)]

neg n s = [(x, y) | (x, y) <- s, x == n, y < 0]

-------------------------------------

--28
-- Zenbaki osozko bikoteez osatutako s zerrenda emanda, bikoteen osagaiei 1
-- balioa gehituz lortzen diren bikoteak dituen zerrenda itzultzen duen
-- funtzioa.

inkrb:: [(Integer, Integer)] -> [(Integer, Integer)]
```

```
inkrb s = [(x + 1, y + 1) | (x, y) <- s]

-------------------------------------

--29
-- Osoa den x zenbaki bat eta zenbaki osoz osatutako s zerrenda bat emanda, s
-- zerrendan dauden x zenbakiaren errepikapenez osatutako zerrenda itzultzen
-- duen funtzioa.

rep:: Integer -> [Integer] -> [Integer]

rep x s = [y | y <- s, y == x]

-------------------------------------
```