```
module Murgilketa where
import Zerrendak2
import Zerrendak

-- Funtzio honek, x eta bm bi zenbaki oso emanda, [bm..x]
-- tartekoak diren x zenbakiaren zatitzaileak itzuliko ditu.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.
-- bm zenbakia 0 edo negatiboa baldin bada, errorea.

zatizer_lag:: Int -> Int -> [Int]
zatizer_lag x bm
        | x < 0 || x == 0       = error "Zenbakia ez da positiboa."
        | bm <= 0               = error "Beheko muga ez da positiboa."
        | bm > x                = []
        | bm > (x `div` 2)      = [x]
        | x `mod` bm == 0       = bm : (zatizer_lag x (bm + 1))
        | otherwise             = zatizer_lag x (bm + 1)
-------------------------------------------------------------------

-- Funtzio honek, x zenbaki oso bat emanda, [1..x]
-- tartekoak diren x zenbakiaren zatitzaileak itzuliko ditu
-- murgilketa erabiliz.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.

zatizer:: Int -> [Int]
zatizer x = zatizer_lag x 1
-------------------------------------------------------------------

-- Funtzio honek, x zenbaki oso bat emanda,
-- lehena al den erabakiko du murgilketa erabili gabe.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.

lehena_gb:: Int -> Bool
lehena_gb x
        | x <= 0                        = error "Zenbakia ez da positiboa."
        | luzera (zatizer x) == 2       = True
        | luzera (zatizer x) /= 2       = False
-------------------------------------------------------------------
-- Funtzio honek, x eta bm bi zenbaki oso emanda, True itzuliko du
-- x zenbakiak [bm..x - 1] tartean zatitzailerik ez badu eta bestela
-- False itzuliko du.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.
-- bm zenbakia 1 edo txikiagoa baldin bada, errorea.
-- Kasu berezi bezala, x zenbakiaren balioa 1 denean False itzuliko du.

lehena_lag:: Int -> Int -> Bool
lehena_lag x bm
        | x < 0 || x == 0       = error "Zenbakia ez da positiboa."
        | bm <= 1               = error "Beheko muga 2 baino txikiagoa."
        | x == 1                = False
        | bm > (x `div` 2)      = True
        | x `mod` bm == 0       = False
        | otherwise             = lehena_lag x (bm + 1)
-------------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- lehena al den erabakiko du [2..x - 1] tartean zatitzailerik ba al
-- duen aztertuz eta murgilketa erabiliz.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.

lehena:: Int -> Bool
lehena x = lehena_lag x 2
-------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat
-- emanda, zerrendan elkarren jarraian dauden elementu berdinez
-- osatutako lehenengo azpizerrendaren luzera itzuliko du.
-- Sarrerako zerrenda hutsa baldin bada, errore-mezua aurkeztuko da.

berdinak_zenbatu:: [Int] -> Int
```

```
berdinak_zenbatu [] = error "Zerrenda hutsa."
berdinak_zenbatu (x:s)
        | hutsa_da s            = 1
        | x == (leh s)          = 1 + (berdinak_zenbatu s)
        | x /= (leh s)          = 1

-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat
-- emanda, zerrendan elkarren jarraian dauden elementu berdinez
-- osatutako lehenengo azpizerrenda kenduz geratzen den zerrenda itzuliko du.
-- Sarrerako zerrenda hutsa baldin bada, errore-mezua aurkeztuko da.

berdinak_kendu:: [Int] -> [Int]

berdinak_kendu [] = error "Zerrenda hutsa."
berdinak_kendu (x:s)
        | hutsa_da s            = []
        | x == (leh s)          = berdinak_kendu s
        | x /= (leh s)          = s

-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda
-- bat emanda, zerrendan elkarren jarraian dauden elementu berdinez
-- osatutako azpizerrenden luzerez eratutako zerrenda itzuliko du.
-- Kalkulua murgilketarik gabe egiten du.
-- Sarrerako zerrenda hutsa baldin bada, zerrenda hutsa itzuliko da.

azpiluz_gb:: [Int] -> [Int]

azpiluz_gb [] = []

azpiluz_gb (x:s) = (berdinak_zenbatu (x:s)):(azpiluz_gb (berdinak_kendu (x:s)))

-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat eta zenbaki oso
-- bat emanda, zerrendan elkarren jarraian dauden elementu berdinez
-- osatutako azpizerrenden luzerez eratutako zerrenda itzuliko du.
-- Lehenengo azpizerrendaren kasuan, bere luzera gehi luz itzuliko du.
-- Sarrerako zerrenda hutsa baldin bada, zerrenda hutsa itzuliko da.

azpiluz_lag:: [Int] -> Int -> [Int]

azpiluz_lag [] luz = []

azpiluz_lag (x:s) luz
        | hutsa_da s            = (1 + luz):[]
        | x == (leh s)          = azpiluz_lag s (1 + luz)
        | x /= (leh s)          = (1 + luz): (azpiluz_lag s 0)

-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat emanda,
-- elkarren jarraian dauden elementu berdinez
-- osatutako azpizerrenden luzerez eratutako zerrenda itzuliko du.
-- Sarrerako datua hutsa baldin bada, zerrenda hutsa itzuliko da.
-- Definizio hau murgilketaren teknikan oinarrituta dago.

azpiluz:: [Int] -> [Int]
azpiluz r = azpiluz_lag r 0

-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat
-- emanda, zerrendan elkarren jarraian dauden elementu berdinez
-- osatutako lehenengo azpizerrenda itzuliko du.
-- Sarrerako zerrenda hutsa baldin bada, errore-mezua aurkeztuko da.

berdinak_hartu:: [Int] -> [Int]
```

```
berdinak_hartu [] = error "Zerrenda hutsa."
berdinak_hartu (x:s)
        | hutsa_da s            = [x]
        | x == (leh s)          = x:(berdinak_hartu s)
        | x /= (leh s)          = [x]

-----------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda
-- bat emanda, zerrendan elkarren jarraian dauden elementu berdinez
-- osatutako azpizerrendez eratutako zerrenda itzuliko du.
-- Kalkulua murgilketarik gabe egiten du.
-- Sarrerako zerrenda hutsa baldin bada, zerrenda hutsa itzuliko da.

azpizer_gb:: [Int] -> [[Int]]

azpizer_gb [] = []

azpizer_gb (x:s) = (berdinak_hartu (x:s)):(azpizer_gb (berdinak_kendu (x:s)))


-----------------------------------------------------------------
-- Funtzio honek, zenbaki osozko bi zerrenda emanda, zerrendan
-- elkarren jarraian dauden elementu berdinez
-- osatutako azpizerrendez eratutako zerrenda itzuliko du.
-- Lehenengo azpizerrendaren kasuan, azpizerrenda hori bigarren
-- parametroarekin elkartuta itzuliko du.
-- Sarrerako lehenengo zerrenda hutsa baldin bada, zerrenda hutsa
-- itzuliko da.

azpizer_lag:: [Int] -> [Int] -> [[Int]]

azpizer_lag [] azpi = []

azpizer_lag (x:s) azpi
        | hutsa_da s            = (x:azpi):[]
        | x == (leh s)          = azpizer_lag s (x:azpi)
        | x /= (leh s)          = (x:azpi):(azpizer_lag s [])

-----------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat emanda, zerrendan
-- elkarren jarraian dauden elementu berdinez
-- osatutako azpizerrendez eratutako zerrenda itzuliko du.
-- Sarrerako lehenengo zerrenda hutsa baldin bada, zerrenda hutsa
-- itzuliko da.
-- Definizio hau murgilketaren teknikan oinarrituta dago.

azpizer:: [Int] -> [[Int]]
azpizer r = azpizer_lag r []

-----------------------------------------------------------------
-----------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- betea al den erabakiko du zerrenda bateko elementuak batzen dituen
-- "batu" eta zenbaki oso baten zatitzaileen zerrenda kalkulatzen duen
-- "zatizer" funtzioak erabiliz.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.
-- Funtzio hau murgilketa erabili gabe definitu da.

betea_gb:: Int -> Bool
betea_gb x
        | x <= 0                                = error "Zenbakia ez da
positiboa."
        | ((batu (zatizer x)) - x) == x         = True
        | otherwise                             = False
-----------------------------------------------------------------
-- Funtzio honek, x, bm eta zatibatu hiru zenbaki oso emanda, True itzuliko du
-- x zenbakiak [bm..x - 1] tartean dituen zatitzaileen batura gehi zatibatu
-- balioa x balioaren berdina al den erabakiko du. ez badu eta bestela
```

```
-- False itzuliko du.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.
-- bm zenbakia 0 edo negatiboa baldin bada, errorea.

betea_lag:: Int -> Int -> Int -> Bool
betea_lag x bm zatibatu
        | x <= 0                                = error "Zenbakia ez da
positiboa."
        | bm <= 0                               = error "Beheko muga ez da
positiboa."
        | (bm > (x `div` 2)) && (zatibatu == x) = True
        | (bm > (x `div` 2)) && (zatibatu /= x) = False
        | (x `mod` bm) == 0                     = betea_lag x (bm + 1) (bm +
zatibatu)
        | (x `mod` bm) /= 0                     = betea_lag x (bm + 1) zatibatu

-----------------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- betea al den erabakiko du murgilketa erabiliz.
-- x zenbakia 0 edo negatiboa baldin bada, errorea.

betea:: Int -> Bool
betea x = betea_lag x 1 0

-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat emanda,
-- zerrendako elementuen batura kalkulatuko du
-- murgilketaren teknika erabili gabe.

batu_gb :: [Int] -> Int

batu_gb [] = 0

batu_gb (x:s) = x +  (batu_gb s)


-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat eta b zenbaki oso bat emanda,
-- zerrendako elementuen batura gehi b kalkulatuko du.

batu_lag :: [Int] -> Int -> Int

batu_lag [] b = b

batu_lag (x:s) b = batu_lag s (x + b)
-----------------------------------------------------------------------
-- Funtzio honek, zenbaki osozko zerrenda bat emanda,
-- zerrendako elementuen batura kalkulatuko du
-- murgilketaren teknika erabiliz.

batu_mr :: [Int] -> Int

batu_mr r = batu_lag r 0


-----------------------------------------------------------------------
-- Funtzio honek, zerrenda bat emanda, bere alderantzizkoa itzuliko
-- du ++ erabiliz eta murgilketa erabili gabe.

alder_gb :: Show t => [t] -> [t]

alder_gb [] = []

alder_gb (x:s) = (alder_gb s) ++ [x]
-----------------------------------------------------------------------
-- Funtzio honek, bi zerrenda emanda, lehenengo zerrendaren
-- alderantzizkoa kalkulatu eta
-- bigarren zerrendari elkartuta itzuliko  du.
```

```
alder_lag :: Show t => [t] -> [t] -> [t]

alder_lag [] q = q

alder_lag (x:s) q = alder_lag s (x:q)
-----------------------------------------------------------------
-- Funtzio honek, zerrenda bat emanda, bere alderantzizkoa itzuliko
-- du murgilketa erabiliz.

alder_mr :: Show t => [t] -> [t]

alder_mr r = alder_lag r []

-----------------------------------------------------------------
-- Funtzio honek, elementu bat eta zerrenda bat emanda, elementu horren
-- agerpen denak kenduz gelditzen den zerrenda itzuliko du
-- murgilketa erabili gabe.


kendu_gb :: (Show t, Eq t) => t -> [t] -> [t]

kendu_gb x [] = []

kendu_gb x (y:s)
        | x == y                = kendu_gb x s
        | x /= y                = y:(kendu_gb x s)

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

-- Funtzio honek, elementu bat eta bi zerrenda emanda, elementu horren
-- agerpen denak lehenengo zerrendatik kenduz gelditzen den zerrenda
-- eta aurretik (ezkerretik) bigarren zerrendaren alderantzizkoa
-- elkartuz lortzen den zerrenda itzuliko du.
-- Aurretik definitutako alder_mr funtzioa erabiltzen da.


kendu_lag_ald :: (Show t, Eq t) => t -> [t] -> [t] -> [t]

kendu_lag_ald x [] q = alder_mr q

kendu_lag_ald x (y:s) q
        | x == y                = kendu_lag_ald x s q
        | x /= y                = kendu_lag_ald x s (y:q)


-----------------------------------------------------------------
-- Funtzio honek, elementu bat eta zerrenda bat emanda, elementu horren
-- agerpen denak kenduz gelditzen den zerrenda itzuliko du
-- zerrenda baten alderantzizko kalkulatzen duen funtzioan oinarritutako
-- murgilketa erabiliz.

kendu_mr_ald :: (Show t, Eq t) => t -> [t] -> [t]

kendu_mr_ald x r = kendu_lag_ald x r []

-----------------------------------------------------------------
-- Funtzio honek, elementu bat eta bi zerrenda emanda, elementu horren
-- agerpen denak lehenengo zerrendatik kenduz gelditzen den zerrenda
-- eta aurretik (ezkerretik) bigarren zerrenda
-- elkartuz lortzen den zerrenda itzuliko du.
-- ++ funtzioa erabiliko da.

kendu_lag_elk :: (Show t, Eq t) => t -> [t] -> [t] -> [t]

kendu_lag_elk x [] q = q

kendu_lag_elk x (y:s) q
```

```
        | x == y                 = kendu_lag_elk x s q
        | x /= y                 = kendu_lag_elk x s (q ++ [y])

-----------------------------------------------------------------------
-- Funtzio honek, elementu bat eta zerrenda bat emanda, elementu horren
-- agerpen denak kenduz gelditzen den zerrenda itzuliko du
-- bi zerrenda elkartzen dituen ++ eragiketan oinarritutako murgilketa
-- erabiliz.

kendu_mr_elk :: (Show t, Eq t) => t -> [t] -> [t]

kendu_mr_elk x r = kendu_lag_elk x r []

-----------------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- bere fibonaccia kalkulatuko du murgilketa erabili gabe.
-- x zenbakia negatiboa baldin bada, errorea.


fib:: Int -> Int
fib x
        | x <= (-1)             = error "Zenbaki negatiboa."
        | x == 0                = 0
        | x == 1                = 1
        | otherwise             = (fib (x - 1)) + (fib (x - 2))

-----------------------------------------------------------------------
-- Funtzio honek, x, bm, aa eta aur lau zenbaki oso emanda,
-- [bm..x] tartea zeharkatuko du.
-- x zenbakia negatiboa baldin bada, errorea.
-- bm-ren balioa 1 edo txikiagoa baldin bada, errorea.

fib_lag:: Int -> Int -> Int -> Int -> Int
fib_lag x bm aa aur
        | x <= (-1)             = error "Zenbaki negatiboa."
        | bm <= 1               = error "Beheko muga 2 baino txikiagoa."
        | x == 0 || x == 1      = x
        | bm > x                = error "Eremu hutsa."
        | bm == x               = aa + aur
        | otherwise             = fib_lag x (bm + 1) aur (aa + aur)

-----------------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- bere fibonaccia kalkulatuko du murgilketa erabiliz.
-- x zenbakia negatiboa baldin bada, errorea.

fib_mr:: Int -> Int

fib_mr x = fib_lag x 2 0 1

-----------------------------------------------------------------------
-- FIBONACCIA INT MOTA ERABILI BEHARREAN INTEGER MOTA ERABILIZ
-----------------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- bere fibonaccia kalkulatuko du murgilketa erabili gabe.
-- x zenbakia negatiboa baldin bada, errorea.
-- Int mota erabili beharrean Integer mota erabili da.

fib2:: Integer -> Integer
fib2 x
        | x <= (-1)             = error "Zenbaki negatiboa."
        | x == 0                = 0
        | x == 1                = 1
        | otherwise             = (fib2 (x - 1)) + (fib2 (x - 2))

-----------------------------------------------------------------------
-- Funtzio honek, x, bm, aa eta aur lau zenbaki oso emanda,
-- [bm..x] tartea zeharkatuko du.
```

```
-- x zenbakia negatiboa baldin bada, errorea.
-- bm-ren balioa 1 edo txikiagoa baldin bada, errorea.
-- Int mota erabili beharrean Integer mota erabili da.

fib2_lag:: Integer -> Integer -> Integer -> Integer -> Integer
fib2_lag x bm aa aur
        | x <= (-1)             = error "Zenbaki negatiboa."
        | bm <= 1               = error "Beheko muga 2 baino txikiagoa."
        | x == 0 || x == 1      = x
        | bm > x                = error "Eremu hutsa."
        | bm == x               = aa + aur
        | otherwise             = fib2_lag x (bm + 1) aur (aa + aur)

-----------------------------------------------------------------------
-- Funtzio honek, x zenbaki oso bat emanda,
-- bere fibonaccia kalkulatuko du murgilketa erabiliz.
-- x zenbakia negatiboa baldin bada, errorea.
-- Int mota erabili beharrean Integer mota erabili da.

fib2_mr:: Integer -> Integer

fib2_mr x = fib2_lag x 2 0 1

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