```
module Murgilketa2 where
import Zerrendak2
import Zerrendak
import Murgilketa

--C ATALEKO ARIKETEN SOLUZIOAK

--1
-- Funtzio honek x eta bm bi zenbaki oso emanda, x zenbakiaren [bm..x]
tarteko
-- faktore lehenen zerrenda itzuliko du (errepikapenik gabe).
-- Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua
-- aurkeztuko da eta bm balioa 2 baino txikiagoa baldin bada
-- ere, errore-mezua aurkeztuko da.

faktoreak_lag:: Int -> Int -> [Int]
faktoreak_lag x bm
        | x < 2                                 = error "Lehenengo datua 2
baino txikiagoa da."
        | bm < 2                                = error "Bigarren datua 2
baino txikiagoa da."
        | bm > x                                = []
        | lehena x                              = [x]
        | (lehena bm) && (x `mod` bm == 0)      = bm : (faktoreak_lag x (bm
+ 1))
        | otherwise                             = faktoreak_lag x (bm + 1)
-----------------------------------------------------------------

-- Funtzio honek 2 baino handiagoa edo berdina den x zenbakiaren
-- errepikapenik gabeko faktore lehenen zerrenda kalkulatuko du
murgilketaren
-- teknika erabiliz. Zenbaki bat lehena al den erabakitzeko, aurretik
-- definituta dagoen lehena izeneko funtzioa erabiliko da.
-- Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztuko
da.

faktoreak:: Int -> [Int]
faktoreak x
        | x < 2         = error "Datua 2 baino txikiagoa da."
        | otherwise     = faktoreak_lag x 2

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

--2
-- Funtzio honek x eta bm bi zenbaki oso emanda, x zenbakiaren [bm..x]
-- tarteko faktore lehenen zerrenda itzuliko du (errepikapenekin).
-- Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztuko

-- da eta bm balioa 2 baino txikiagoa baldin bada ere, errore-mezua
-- aurkeztuko da.

desk_lag:: Int -> Int -> [Int]
desk_lag x bm
        | x < 2                                 = error "Lehenengo datua 2
baino txikiagoa da."
        | bm < 2                                = error "Bigarren datua 2
baino txikiagoa da."
        | bm > x                                = []
        | lehena x                              = [x]
        | (lehena bm) && (x `mod` bm == 0)      = bm : (desk_lag (x `div`
bm) bm)
        | otherwise                             = desk_lag x (bm + 1)
-----------------------------------------------------------------

-- Funtzio honek 2 baino handiagoa edo berdina den x zenbakiaren
-- errepikapendun faktore lehenen zerrenda kalkulatuko du murgilketaren
-- teknika erabiliz. Zenbaki bat lehena al den erabakitzeko, aurretik
-- definituta dagoen lehena izeneko funtzioa erabiliko da.
-- Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztuko
da.

desk:: Int -> [Int]
```

```
desk x
        | x < 2         = error "Datua 2 baino txikiagoa da."
        | otherwise     = desk_lag x 2

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

--3
-- Funtzio honek n eta bm bi zenbaki oso emanda,
-- bm-tik abiatuta lehenengo n zenbaki lehenez osatutako zerrenda
-- kalkulatuko du. Emandako n balioa 0 baino txikiagoa baldin bada,
-- errore-mezua aurkeztuko da eta bm balioa 2 baino txikiagoa baldin
-- bada ere, errore-mezua aurkeztuko da.

leh_zerrenda_lag:: Int -> Int -> [Int]
leh_zerrenda_lag n bm
        | n < 0                         = error "Lehenengo datua 0 baino
txikiagoa da."
        | bm < 2                        = error "Bigarren datua 2 baino
txikiagoa da."
        | n == 0                        = []
        | (lehena bm)                   = bm : (leh_zerrenda_lag (n - 1)
(bm + 1))
        | otherwise                     = leh_zerrenda_lag n (bm + 1)
-----------------------------------------------------------------

-- Funtzio honek lehenengo n zenbaki lehenez osatutako zerrenda kalkulatuko
du
-- murgilketaren teknika erabiliz. n-ren balioa 0 baino txikiagoa bada,
-- errore-mezua aurkeztuko da. Zenbaki bat lehena al den erabakitzeko,
-- aurretik definituta dagoen lehena izeneko funtzioa erabiliko da.

leh_zerrenda:: Int -> [Int]
leh_zerrenda n
        | n < 0         = error "Datua 0 baino txikiagoa da."
        | otherwise     = leh_zerrenda_lag n 2

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

--4
-- Funtzio honek n eta bm bi zenbaki oso emanda,
-- bm-ren berdinak edo handiagoak eta n baino txikiagoak diren zenbaki
-- lehenez osatutako zerrenda kalkulatuko du.
-- Emandako n balioa 2 baino txikiagoa baldin bada, errore-mezua
-- aurkeztuko da eta bm balioa 2 baino txikiagoa
-- baldin bada ere, errore-mezua aurkeztuko da.

tx_zerrenda_lag:: Int -> Int -> [Int]
tx_zerrenda_lag n bm
        | n < 2                         = error "Lehenengo datua 2 baino
txikiagoa da."
        | bm < 2                        = error "Bigarren datua 2 baino
txikiagoa da."
        | n <= bm                       = []
        | (lehena bm)                   = bm : (tx_zerrenda_lag n (bm + 1))
        | otherwise                     = tx_zerrenda_lag n (bm + 1)
-----------------------------------------------------------------

-- Funtzio honek n zenbakia baino txikiagoak diren zenbaki lehenez
-- osatutako zerrenda kalkulatuko du murgilketaren teknika erabiliz.
-- n-ren balioa >= 2 ez bada, errore-mezua aurkeztuko da.
-- Zenbaki bat lehena al den erabakitzeko, aurretik definituta dagoen
-- lehena izeneko funtzioa erabiliko da.

tx_zerrenda:: Int -> [Int]
tx_zerrenda n
        | n < 0         = error "Datua 2 baino txikiagoa da."
        | otherwise     = tx_zerrenda_lag n 2

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

--5
-- Funtzio honek x eta p bi zenbaki oso emanda, x = p * (2^k)
```

```
-- betearazten duen eta 0 edo handiagoa den k zenbaki osorik
-- ba al den ala ez erabaki beharko du.
-- Emandako x balioa 1 baino txikiagoa baldin bada, errore-mezua
-- aurkeztu beharko da eta p balioa 1 baino txikiagoa baldin bada
-- ere, errore-mezua aurkeztu beharko da.

-- biber_lag x p kalkulatzeko honakoa egin behar da:
-- p parametroan 2 zenbakia bidertuz joan behar da, 2 behin eta
-- berriz bidertuz x balioa lortzerik ba al dagoen jakiteko. x
-- eta p-ren balioak berdinak izatea lortuz gero, True erantzunez
-- bukatuko da. Baina p-ren balioa x baino handiagoa izatera iristen
-- bada, x eta p berdinak izateko aukerarik ez dagoenez False itzuliko da.

biber_lag:: Int -> Int -> Bool
biber_lag x p
        | x < 1                         = error "Lehenengo datua 1 baino
txikiagoa da."
        | p < 1                         = error "Bigarren datua 1 baino
txikiagoa da."
        | x < p                         = False
        | x == p                        = True
        | otherwise                     = biber_lag x (2 * p)
-----------------------------------------------------------------------

-- Funtzio honek datu bezala x zenbaki oso bat emanda, x balioa 2
-- zenbakiaren berredura al den erabakiko du murgilketa erabiliz.
-- x zenbakiaren balioa 0 edo txikiagoa baldin bada, errore-mezua
-- aurkeztuko da.
-- 0 baino handiagoa den zenbaki oso bat 2ren berredura dela
-- esaten da, x = 2^k betearazten duen eta 0 edo handiagoa den k
-- zenbaki osorik baldin bada.

biber:: Int -> Bool
biber x
        | x < 1         = error "Datua 1 baino txikiagoa da."
        | otherwise     = biber_lag x 2

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

--6
-- x eta v bi zenbaki oso, l zenbaki osozko zerrenda
-- eta zenb zenbaki oso bat emanda, x zenbakiaren l zerrendako
-- agerpen-kopurua gehi zenb balioa v baino handiagoa bada
-- True eta bestela False itzuliko du funtzio honek.


-- gehiagotan_lag x v l zenb kalkulatzeko urratsak honako hauek dira:
--      * zenb balioa v baino handiagoa bada, True itzuli emaitza bezala.
--      * Bestela, l hutsa baldin bada, (eta zenb ez da v baino handiagoa),

--        False itzuli emaitza bezala.
--      * Bestela, l-ko lehenengo elementua x balioaren berdina bada,
hurrengo
--        dei errekurtsiboan x eta v berdin mantenduko dira, l-ren lekuan
l-ko
--        ezkerreko ertzeko elementua kenduz lortzen den zerrenda ipini
beharko
--        da eta zenb-en ordez zenb + 1 ipini beharko da.
--      * Bestela, l-ko lehenengo elementua x balioaren berdina ez bada,
l-ren
--        lekuan l-ko ezkerreko ertzeko elementua kenduz lortzen den
zerrenda
--        ipini beharko da eta zenb berdin mantenduko da.

gehiagotan_lag:: Int -> Int -> [Int] -> Int -> Bool
gehiagotan_lag x v l zenb
        | zenb > v              = True
        | l == []               = False
        | (leh l) == x          = gehiagotan_lag x v (hond l) (zenb + 1)
        | otherwise             = gehiagotan_lag  x v (hond l) zenb
-----------------------------------------------------------------------
```

```
-- Datu bezala x eta v bi zenbaki oso eta zenbaki osozko l zerrenda bat
emanda,
-- x zenbakiaren l zerrendako agerpen-kopurua v baino handiagoa al den
-- erabakiko du funtzio honek.

gehiagotan:: Int -> Int -> [Int] -> Bool
gehiagotan x v l = gehiagotan_lag x v l 0

-- gehiagotan x v l definitzeko gehiagotan_lag erabiltzerakoan, zenb
-- parametroa 0 balioaz ordezkatuko da, zenbatzaile bezala erabiliko baita.

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

--7
-- Funtzio honek zenbaki osozko s zerrenda bat eta h zenbaki oso bat
-- emanda, s zerrendako elementuak eta h balioa hartuz, handiena
-- itzuliko du.
-- Emandako s zerrenda hutsa baldin bada, h izango da emaitza.


-- hand_mr_lag s h kalkulatzeko honakoa egin behar da:
--      * s hutsa baldin bada, emaitza bezala h itzuli.
--      * s hutsa ez bada, s-ko lehenengo elementua h-rekin konparatu eta
--        hurrengo dei errekurtsiborako h eta s-ko lehenengoa hartuz,
bietako
--        handiena ipini behar da h-ren lekuan. Prozesua, zerrenda hutsa
--        geratzen denean bukatuko da.

hand_mr_lag:: [Int] -> Int -> Int
hand_mr_lag l h
        | l == []               = h
        | (leh l) <= h          = hand_mr_lag (hond l) h
        | otherwise             = hand_mr_lag (hond l) (leh l)
-----------------------------------------------------------------------

-- Datu bezala zenbaki osozko l zerrenda bat emanda, l zerrendako
-- balio handiena itzultzen duen hand_mr funtzioa definitu.
-- l zerrenda hutsa baldin bada, errore-mezua aurkeztu beharko da.

hand_mr:: [Int] -> Int
hand_mr l
        | hutsa_da l            = error "Zerrenda hutsa"
        | otherwise             = hand_mr_lag (hond l) (leh l)

-- hand_mr_lag funtzioa erabiliz hand_mr l definitzeko, l hutsa baldin
bada,
-- errore-mezua aurkeztuko da eta l hutsa ez bada, hand_mr_lag funtzioari
-- deituko zaio s-ren lekuan l-tik lehenengo elementua kenduz geratzen den
-- zerrenda eta h-ren lekuan l-ko lehenengo elementua ipiniz.

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

--8
-- Funtzio honek zenbaki osozko l zerrenda bat eta bik eta bak bi
-- zenbaki oso emanda, l zerrendako bikoiti-kopurua gehi bik balioa
-- l zerrendako bakoiti-kopurua gehi bak balioa baino handiagoa baldin
-- bada True eta bestela False itzuliko du.

bg_lag:: [Int] -> Int -> Int -> Bool
bg_lag l bik bak
        | (l == []) && bik > bak        = True
        | (l == []) && bik <= bak       = False
        | (leh l) `mod` 2 == 0          = bg_lag (hond l) (bik + 1) bak
        | otherwise                     = bg_lag (hond l) bik (bak + 1)
-----------------------------------------------------------------------

-- Funtzio honek, murgilketaren teknika erabiliz, zenbaki osoz osatutako
-- l zerrenda bat emanda, l zerrendan bikoiti-kopurua bakoiti-kopurua
-- baino handiagoa baldin bada True eta bestela False itzuliko du.
-- l zerrenda hutsa baldin bada, False itzuliko da.

bg:: [Int] -> Bool
```

```
bg l = bg_lag l 0 0

-- bg_lag erabiliz bg l definitzerakoan, bik eta bak parametroak zenbaki
-- bikoitiak eta bakoitiak zenbatzeko erabiliko dira hurrenez hurren.

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