```
module Zerrendak2 where

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

import Ez_errek {- bikoitia eta bakoitia funtzioak
                  erabili ahal izateko -}
import Zerrendak {- leh, hond, hutsa_da, badago eta luzera
                    erabili ahal izateko -}

-------------------------------------------------------
--1
-- Funtzio honek, zenbaki osozko zerrenda bat emanda,
-- zerrendako elementu bakoitzari 1 balioa gehituz lortzen
-- den zerrenda itzuliko du.
-- Emandako zerrenda hutsa baldin bada, zerrenda hutsa
-- itzuliko du.

inkr:: [Int] -> [Int]
inkr [] = []
inkr (x:s) = (x + 1): (inkr s)
-------------------------------------------------------
--2
-- Funtzio honek, zenbaki osozko zerrenda bat emanda,
-- zerrendako elementuen batura kalkulatuko du.
-- Emandako zerrenda hutsa baldin bada, 0 itzuliko du.

batu:: [Int] -> Int
batu [] = 0
batu (x:s) = x + (batu s)

-------------------------------------------------------
--3
-- Funtzio honek, zenbaki osozko zerrenda bat emanda,
-- zerrendan balio bikoitirik ba al dagoen erabakiko du.

bikoitirik:: [Int] -> Bool
bikoitirik [] = False
bikoitirik (x:s)
        | bikoitia x    = True
        | bakoitia x    = bikoitirik s

-------------------------------------------------------
--4
-- Funtzio honek, zenbaki osozko bi zerrenda emanda,
-- posizio bereko elementuak biak bikoitiak edo biak
-- bakoitiak badira kasu denetan, orduan True itzuliko du
-- eta bestela False.
```

```
-- Zerrendak luzera desberdinekoak badira, errore-mezua
-- aurkeztuko da.

bik_posi:: [Int] -> [Int] -> Bool
bik_posi [] r
        |not (hutsa_da r)       = error "Luzera desberdineko zerrendak."
        |otherwise              = True
bik_posi (x:s) r
        |(luzera (x:s)) /= (luzera r)           = error "Luzera desberdineko zerrendak."
        |(bikoitia x) && (bakoitia (leh r))     = False
        |(bakoitia x) && (bikoitia (leh r))     = False
        |otherwise                              = bik_posi s (hond r)

-----------------------------------------------------
--5
-- Funtzio honek, zerrenda bat emanda,
-- zerrendako azkeneko elementua itzuliko du.
-- Emandako zerrenda hutsa baldin bada, errore-mezua
-- aurkeztuko du.

azkena:: [t] -> t
azkena [] = error "Zerrenda hutsa."
azkena (x:s)
        | hutsa_da s            = x
        | otherwise             = azkena s

-----------------------------------------------------
--6
-- Funtzio honek, zerrenda bat emanda,
-- zerrendako azkeneko elementua kenduz gelditzen den
-- zerrenda itzuliko du.
-- Emandako zerrenda hutsa baldin bada, errore-mezua
-- aurkeztuko du.

azkena_kendu:: [t] -> [t]
azkena_kendu [] = error "Zerrenda hutsa."
azkena_kendu (x:s)
        | hutsa_da s            = []
        | otherwise             = x: (azkena_kendu s)

-----------------------------------------------------
--7
-- Funtzio honek, zerrenda bat emanda,
-- zerrendako elementuak alderantzizko ordenean ipiniz
-- lortzen den zerrenda itzuliko du.
-- Emandako zerrenda hutsa baldin bada, zerrenda hutsa
-- itzuliko du.
```

```
alder:: [t] -> [t]
alder [] = []
alder (x:s) = (alder s) ++ (x:[])

-- (BIGARREN AUKERA) Funtzio honek, zerrenda bat emanda,
-- zerrendako elementuak alderantzizko ordenean ipiniz
-- lortzen den zerrenda itzuliko du.
-- Emandako zerrenda hutsa baldin bada, zerrenda hutsa
-- itzuliko du.

alder2:: [t] -> [t]
alder2 [] = []
alder2 (x:s) = (azkena (x:s)) : (alder2 (azkena_kendu (x:s)))

-----------------------------------------------------
--8
-- Funtzio honek, bi zerrenda emanda, bata bestearen
-- alderantzizkoa al den erabakiko du.

alder_dira:: Eq t => [t] -> [t] -> Bool
alder_dira s r = (s == (alder r))


-- (BIGARRREN AUKERA) Funtzio honek, bi zerrenda emanda,
-- bata bestearen alderantzizkoa al den erabakiko du.

alder_dira2:: Eq t => [t] -> [t] -> Bool
alder_dira2 [] r
        |not (hutsa_da r)       = False
        |otherwise              = True
alder_dira2 (x:s) r
        |(luzera (x:s)) /= (luzera r)   = False
        |x /= (azkena r)                = False
        |x == (azkena r)                = alder_dira2 s (azkena_kendu r)

-----------------------------------------------------
-----------------------------------------------------------
--9
-- Elementu bat eta zerrenda bat emanda, elementu hori
-- zerrendan zenbat aldiz agertzen den erabakitzen duen funtzioa

aldiz:: Eq t => t -> [t] -> Int
aldiz x [] = 0
aldiz x (y:s)
        | x == y                = 1 + (aldiz x s)
        | otherwise             = aldiz x s
```

```
------------------------------------------------------
--10
-- Zerrenda batean errepikatutako elementurik ba al dagoen
-- erabakitzen duen funtzioa

erre:: Eq t => [t] -> Bool
erre [] = False
erre (x:s)
        | badago x s            = True
        | otherwise             = erre s

------------------------------------------------------
--11
-- Elementu bat eta zerrenda bat emanda, elementu horren
-- agerpen denak kenduz gelditzen den zerrenda itzultzen
-- duen funtzioa

kendu:: Eq t => t -> [t] -> [t]
kendu x [] = []
kendu x (y:s)
        | x == y                = kendu x s
        | otherwise             = y:(kendu x s)

------------------------------------------------------
--12
-- Zenbaki osoz osatutako zerrenda bat emanda, elementu
-- bikoiti denak kenduz lortzen den zerrenda itzultzen duen
-- funtzioa

bik_kendu:: [Int] -> [Int]
bik_kendu [] = []
bik_kendu (x:s)
        | bikoitia x            = bik_kendu s
        | bakoitia x            = x:(bik_kendu s)


{-Ondorengo bi funtzioak Zerrendak.hs moduluan daude
------------------------------------------------------
--13
-- Zerrenda bat emanda, posizio bikoitietan dauden elementuak kenduz
-- lortzen den zerrenda itzultzen duen funtzioa

pos_bik_kendu:: [t] -> [t]

pos_bik_kendu [] = []
pos_bik_kendu (x:s)
```

```
        | hutsa_da s            = x:[]
        | otherwise             = x:(pos_bik_kendu (hond s))

------------------------------------------------------------
--14
-- Zerrenda bat emanda, posizio bakoitietan dauden elementuak
-- kenduz lortzen den zerrenda itzultzen duen funtzioa

pos_bak_kendu:: [t] -> [t]

pos_bak_kendu [] = []
pos_bak_kendu (x:s)
        | hutsa_da s            = []
        | otherwise             = (leh s):(pos_bak_kendu (hond s))

------------------------------------------------------------
-}

------------------------------------------------------------
--15
-- Zerrenda bat emanda, errepikapenak kenduz gelditzen den
-- zerrenda itzultzen duen funtzioa.
-- Elementu bakoitzaren kasuan bere lehenengo agerpena lagako du.

erre_kendu:: Eq t => [t] -> [t]
erre_kendu [] = []
erre_kendu (x:s)
        | badago x s            = x:(erre_kendu (kendu x s))
        | otherwise             = x:(erre_kendu s)

------------------------------------------------------------
--16
-- Zerrenda bat emanda, errepikapenak kenduz gelditzen den
-- zerrenda itzultzen duen funtzioa.
-- Elementu bakoitzaren kasuan bere azkenengo agerpena lagako da

erre_kendu2:: Eq t => [t] -> [t]
erre_kendu2 [] = []
erre_kendu2 (x:s)
        | badago x s            = erre_kendu2 s
        | otherwise             = x:(erre_kendu2 s)

------------------------------------------------------------
--17
-- Zerrenda bat emanda, berdinak diren bi elementu jarraian
-- agertzen al diren ala ez erabakitzen duen funtzioa
```

```
bikote_berdin:: Eq t => [t] -> Bool
bikote_berdin [] = False
bikote_berdin (x:s)
        | hutsa_da s            = False
        | x == (leh s)  = True
        | otherwise             = bikote_berdin s

-------------------------------------------------------
--18
-- Zerrenda bat beste zerrenda baten aurrizkia al den
-- erabakitzen duen funtzioa

aurrizkia:: Eq t => [t] -> [t] -> Bool
aurrizkia [] r = True
aurrizkia (x:s) r
        | hutsa_da r            = False
        | x /= (leh r)          = False
        | otherwise             = aurrizkia s (hond r)

-- v zerrenda bat u zerrenda baten aurrizkia izango da
-- v ++ w = u betetzen duen w zerrenda existitzen bada.
-- Horregatik u zerrenda bat hartuta, zerrenda hutsa bere
-- aurrizkia izango da [] ++ u = u betetzen delako.

-------------------------------------------------------
--19
-- Zerrenda bat beste zerrenda baten azpizerrenda al den
-- erabakitzen duen funtzioa

azpizer:: Eq t => [t] -> [t] -> Bool
azpizer [] r = True
azpizer (x:s) r
        | hutsa_da r            = False
        | aurrizkia (x:s) r     = True
        | otherwise             = azpizer (x:s) (hond r)

-- v zerrenda bat u zerrenda baten azpizerrenda izango da
-- x ++ v ++ w = u betetzen duten x eta w bi zerrenda existitzen
-- badira.
-- Horregatik u edozein zerrenda hartuta, zerrenda hutsa bere
-- azpizerrenda izango da [] ++ [] ++ u = u betetzen delako.
-- Hor x zerrenda [] izango da eta w zerrenda u izango da.

-------------------------------------------------------
--20
-- Posizio bat (zenbaki oso bat) eta zerrenda bat emanda,
-- errendako posizio horretan dagoen elementua itzultzen duen
```

```
-- funtzioa.
-- Emandako posizioa tarte egokian ez badago (1 eta zerrendaren
-- luzeraren artean), errore-mezua aurkeztuko da.

elem_pos:: Int -> [t] -> t
elem_pos pos [] = error "Ez da egokia"
elem_pos pos (x:s)
        | (pos < 1) || (pos > luzera (x:s))     = error "Ez da egokia"
        | pos == 1                              = x
        | otherwise                             = elem_pos (pos - 1) s

-----------------------------------------------------------
--21
-- Posizio bat (zenbaki oso bat), elementu bat eta zerrenda bat emanda,
-- zerrendako posizio horretan elementua sartuz lortzen den zerrenda
-- berria itzultzen duen funtzioa. Elementu berria sartzerakoan posizio
-- horretatik aurrera dauden elementuak eskuinera desplazatuta
-- geldituko dira.
-- Emandako posizioa tarte egokian ez badago (1 eta zerrendaren luzera
-- gehi 1en artean), errore-mezua aurkeztuko da.

sartu:: Int -> t -> [t] -> [t]
sartu pos x []
        | pos /= 1      = error "Ez da egokia"
        | otherwise     = x:[]

sartu pos x (y:s)
        | (pos < 1) || (pos > (luzera (x:s) + 1))       = error "Ez da egokia"
        | pos == 1                                      = x:y:s
        | otherwise                                     = y:(sartu (pos - 1) x s)

-----------------------------------------------------------
--22
-- Zenbaki osoz osatutako zerrenda bat emanda, zerrendako
-- lehenengo zenbaki bikoitiaren posizioa itzultzen duen funtzioa.
-- Zenbaki bikoitirik ez badago, zerrendaren luzera gehi 1 itzuliko du.

lehen_bik_pos:: [Int] -> Int
lehen_bik_pos [] = 1

lehen_bik_pos (x:s)
        | x `mod`2 == 0         = 1
        | otherwise             = 1 + (lehen_bik_pos s)

-----------------------------------------------------------
--23
-- Zenbaki oso bat eta zenbaki osoz osatutako zerrenda bat emanda,
```

```
-- zenbaki horren azken agerpenaren posizioa itzultzen duen
-- funtzioa.
-- Zenbakia zerrendan ez bada agertzen, 0 balioa itzuliko da.

azken_pos:: Int -> [Int] -> Int
azken_pos x [] = 0

azken_pos x (y:s)
        | badago x s            = 1 + (azken_pos x s)
        | x == y                = 1
        | otherwise             = 0

------------------------------------------------------
--24
-- Zenbaki osoz osatutako zerrenda bat emanda, zerrendako
-- balio handiena itzultzen duen funtzioa.
-- Zerrenda hutsa bada, errore-mezua aurkeztuko da:

hand:: [Int] -> Int
hand [] = error "Zerrenda hutsa"

hand (x:s)
        | hutsa_da s            = x
        | x <= (leh s)          = hand s
        | otherwise             = hand (x:(hond s))

------------------------------------------------------
--25
--(LEHENENGO AUKERA)

-- t motako bi zerrenda emanda, lehenengo zerrendako lehenengo
-- elementua eta bigarren zerrendako lehenengo elementua ondoan
-- ipiniz zerrendak elkartuz lortzen den zerrenda itzultzen duen
-- funtzioa.
-- Bigarren zerrendako elementuak orden berean geldituko dira baina
-- lehenengo zerrendako elementuak alderantzizko ordenean geldituko dira.

elkartu:: [t] -> [t] -> [t]
elkartu [] r = r

elkartu (x:s) r = elkartu s (x:r)

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

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

--(BIGARREN AUKERA)
```

```
-- t motako bi zerrenda emanda, lehenengo zerrendako lehenengo
-- elementua eta bigarren zerrendako lehenengo elementua ondoan
-- ipiniz zerrendak elkartuz lortzen den zerrenda itzultzen duen
-- funtzioa.
-- Beraz, lehenengo zerrendaren alderantzizkoa eta bigarren
-- zerrenda bat eginez lortzen den zerrenda itzuliko du.

elkartu2:: [t] -> [t] -> [t]
elkartu2 [] r = r

elkartu2 (x:s) r = (alder (x:s)) ++ r

------------------------------------------------------------
--26
-- Zenbaki oso bat eta t motako zerrenda bat emanda, zerrenda horretatik
-- zenbakiak adierazten duen adina elementu ezabatuz lortzen den zerrenda
-- tzultzen duen funtzioa.
-- Elementuak ezkerretik hasita ezabatuko dira.
-- Zenbakia 0 eta zerrendaren luzeraren artean ez badago, errore-mezua
-- aurkeztuko da.
-- Zenbakia 0 bada, ez da elementurik ezabatuko.

ezabatu:: Int -> [t] -> [t]
ezabatu c []
        | c /= 0        = error "Ez da egokia"
        | otherwise     = []

ezabatu c (x:s)
        | (c < 0) || (c > (luzera (x:s)))       = error "Ez da egokia"
        | c == 0                                = x:s
        | otherwise                             = ezabatu (c - 1) s

------------------------------------------------------------
--27
-- Zenbaki oso bat eta t motako zerrenda bat emanda, zerrenda
-- horretatik zenbakiak adierazten duen adina elementu hartuz
-- lortzen den zerrenda itzultzen duen funtzioa.
-- Elementuak ezkerretik hasita hartuko dira.
-- Zenbakia 0 eta zerrendaren luzeraren artean ez badago,
-- errore-mezua aurkeztuko da.
-- Zenbakia 0 bada, zerrenda hutsa itzuliko da.

hartu:: Int -> [t] -> [t]
hartu c []
        | c /= 0        = error "Ez da egokia"
        | otherwise     = []
```

```
hartu c (x:s)
        | (c < 0) || (c > (luzera (x:s)))       = error "Ez da egokia"
        | c == 0                                = []
        | otherwise                             = x:(hartu (c - 1) s)

-------------------------------------------------------
--28
-- Zenbaki osoz osatutako zerrenda bat emanda, jarraian
-- errepikatuta agertzen diren elementuen kopia bakarra
-- lagaz osatzen den zerrenda itzultzen duen funtzioa.
-- Beraz funtzioak jarraian dauden kopiak ezabatuko ditu,
-- kopia bakarra lagaz.
-- Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da.

kolapsatu:: [Int] -> [Int]
kolapsatu [] = []

kolapsatu (x:s)
        | hutsa_da s            = x:[]
        | x /= (leh s)          = x:(kolapsatu s)
        | otherwise             = kolapsatu s

-------------------------------------------------------
--29
-- Zenbaki osoz osatutako zerrenda bat emanda, zenbaki
-- bikoitiak batuketaren bidez elementu bakoiti bat aurkitu arte
-- edo zerrenda bukatu arte eskuinerantz hedatuz lortzen den
-- zerrenda itzultzen duen funtzioa.
-- Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da.

hedatu:: [Int] -> [Int]
hedatu [] = []

hedatu (x:s)
        | hutsa_da s                    = x:[]
        | (x `mod` 2) /= 0              = x:(hedatu s)
        | (leh s) `mod` 2 /= 0          = x:(leh s):(hedatu (hond s))
        | otherwise                     = hedatu ((x + (leh s)):(hond s))

-------------------------------------------------------
--30
-- Zenbaki osoz osatutako zerrenda bat emanda, ezkerretik
-- hasi eta elementu bakoitzak bere atzetik datozen zenbaki
-- txikiagoak (handiagoa edo berdina den bat agertu arte)
-- apalduz (ordezkatuz) lortzen den zerrenda itzultzen duen
-- funtzioa.
```

```
-- Handiagoa edo berdina den zenbaki bat agertzen denean,
-- zenbaki hori hasiko da bere ondoren dauden zenbakiak zapaltzen.
-- Hasierako zerrenda hutsa bada, funtzioak zerrenda
-- hutsa itzuli behar du.

zapaldu:: [Int] -> [Int]
zapaldu [] = []

zapaldu (x:s)
        | hutsa_da s                    = x:[]
        | x <= (leh s)                  = x:(zapaldu s)
        | otherwise                     = x:(zapaldu (x:(hond s)))

-------------------------------------------------------
--31
-- Zenbaki osoz osatutako zerrenda bat emanda, lehenengo elementua bera baino
-- handiagoa edo berdina den elementu bat aurkitu arte hondoratuz (eskuinera
-- desplazatuz) lortzen den zerrenda itzultzen duen hondoratu izeneko funtzioa.
-- Kasu honetan lehenengo elementua baino handiagoa edo berdina den zenbakia
-- aurkitzen denean prozesua bukatu egingo da, eta ondoren dauden elementuak
-- berdin geldituko dira. Hasierako zerrenda hutsa bada, zerrenda hutsa
-- itzuliko da eta hasierako zerrendak elementu bakarra badu, zerrenda
-- hori bera itzuliko da.

hondoratu:: [Int] -> [Int]
hondoratu [] = []

hondoratu (x:s)
        | hutsa_da s            = x:[]
        | x <= (leh s)          = x:s
        | otherwise             = (leh s):(hondoratu (x:(hond s)))

-------------------------------------------------------
--32
-- Zenbaki osoz osatutako zerrenda bat eta eskuinetik kontatzen hasita
-- zerrendako posizio bat (zenbaki oso bat) emanda, posizio horretako
-- elementua kenduz lortzen den zerrenda itzultzen duen funtzioa.
-- Zerrenda hutsa bada, errore-mezu bat aurkeztuko da eta zerrenda
-- hutsa ez bada baina posizioa ez badago 1 eta zerrendaren
-- luzeraren artean, orduan ere errore-mezu bat aurkeztuko da.

elem_kendu:: [Int] -> Int -> [Int]

elem_kendu [] pos = error "Zerrenda hutsa"

elem_kendu (x:s) pos
        | (pos < 1) || (pos > (luzera (x:s)))           = error "Ez da egokia"
```

```
        | pos == (luzera (x:s))                          = s
        | otherwise                                      = x:(elem_kendu s pos)

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

--33
-- Zenbaki osoz osatutako zerrenda bat emanda, zerrendako lehenengo elementua
-- baino txikiagoak diren elementu denak kenduz eta lehenengo elementu hori
-- ukaeran ipiniz lortzen den zerrenda itzultzen duen funtzioa.
-- Emandako zerrenda hutsa bada, errore-mezua itzuliko du funtzioak.

-- Beraz, lehenengo elementuak zerrenda osoa zeharkatuko du bera baino
-- txikiagoak direnak desageraraziz eta bera bukaeran geldituz.

garbitu:: [Int] -> [Int]
garbitu [] = error "Zerrenda hutsa"


garbitu (x:s)
        | hutsa_da s                    = x:[]
        | x > (leh s)                   = garbitu (x:(hond s))
        | otherwise                     = (leh s):(garbitu (x:(hond s)))

-----------------------------------------------------------
--34
-- Zenbaki osoz osatutako zerrenda bat emanda, zerrendako lehenengo elementua
-- behean zehazten dena bete arte eskuinera desplazatuz lortzen den zerrenda
-- itzultzen duen funtzioa.
-- Noiz arte desplazatuko den lehenengo elementua:
--      * Lehenengo elementu hori zerrenda osoan ez bada agertzen,
--        bukaerara iritsi arte desplazatuko da.
--      * Lehenengo elementu hori zerrendan agertzen bada, zenbakia bera
--        agertu arte desplazatuko da eta zenbakiaren kopia biak bi zeroz
--        ordezkatuko dira.
-- Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da.

-- Beraz, lehenengo elementuak zerrenda zeharkatuz joan beharko du bere
-- berdina den elementu bat aurkitu arte edo (ez badago) zerrenda bukatu arte.
-- Berdina den elementua aurkituz gero elementu biak bi zeroz ordezkatuko dira.

zeharkatu:: [Int] -> [Int]
zeharkatu [] = []

zeharkatu (x:s)
        | hutsa_da s            = x:[]
        | x == (leh s)          = 0:0:(hond s)
        | otherwise             = (leh s):(zeharkatu (x:(hond s)))
```

```
------------------------------------------------------------
--35
-- Zenbaki osoz osatutako zerrenda bat emanda, sarrerako zerrendak baino
-- elementu bat gutxiago duen eta jarraian aipatzen diren bi irizpideak
-- jarraituz lortzen den zerrenda itzultzen duen funtzioa:
--      * Zerrendako lehenengo elementuaz zati daitezkeen elementuak
--        (hodarra zero ematen dutenak) lehenengo elementuaz zatituz
--        lortzen den emaitzaz ordezkatu.
--      * Lehenengo elementuaz zati ezin daitezkeenak dauden bezala laga.
--
-- Hasierako zerrenda hutsa bada, errore-mezua aurkeztuko da.
-- Hasierako zerrendak elementu bakarra badu, zerrenda hutsa itzuliko da.

-- Beraz, lehenengo elementuak zerrenda osoa zeharkatuko du zatitu
-- ditzakeen elementuak zatituz eta azkenean bera desagertu egingo da.


zatitu:: [Int] -> [Int]
zatitu [] = error "Zerrenda hutsa"


zatitu (x:s)
        | hutsa_da s                            = []
        | ((leh s) `mod` x) == 0                = ((leh s) `div` x):zatitu (x:(hond s))

        | otherwise                             = (leh s):(zatitu (x:(hond s)))

------------------------------------------------------------
--36
-- Zenbaki osoz osatutako zerrenda bat emanda, zerrendako zenbaki bikoiti denen
-- batura bukaeran ipiniz eta zenbaki bakoitiak mantenduz lortzen den zerrenda
-- itzultzen duen funtzioa.
-- Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da.
-- Hasierako zerrendak elementu bakarra badu, zerrenda bera itzuliko da.

-- Beraz, lehenengo elementu bikoitia eskuinerantz desplazatuz joango da.
-- Zenbaki bikoiti bat aurkitzen badu, bien baturaz ordezkatuko dira bi
-- zenbaki horiek. Baturaren emaitza bezala lortzen den zenbaki bikoiti berri
-- hori hasiko da jarraian eskuinerantz mugitzen.
-- Bakoitiak bere hortan laga behar dira.


superbik:: [Int] -> [Int]
superbik [] = []
```

```
superbik (x:s)
        | hutsa_da s                    = x:[]
        | (x `mod` 2) /= 0              = x:(superbik s)
        | ((leh s) `mod` 2) /= 0        = (leh s):superbik(x:(hond s))
        | otherwise                     = superbik((x + (leh s)):(hond s))

-------------------------------------------------------
--37
-- Zenbaki osoz osatutako zerrenda bat emanda, posizio bakoitzean sarrerako
-- zerrendako lehenengo posiziotik posizio horretara artekoen batura
-- duen zerrenda itzultzen duen funtzioa.
-- Emandako zerrenda hutsa bada, zerrenda hutsa itzuliko du funtzioak.

-- Beraz, zerrenda zeharkatuz eta posizio bakoitzera arte metatutako
-- batura kalkulatuz joango da.


metatu:: [Int] -> [Int]
metatu [] = []


metatu (x:s)
        | hutsa_da s            = x:[]
        | otherwise             = x:metatu((x + (leh s)):(hond s))

-------------------------------------------------------
--38
-- Boolearrez osatutako zerrenda bat eta zenbaki osoz osatutako beste
-- zerrenda bat emanda, zerrenda biak aldi berean zeharkatuz eta jarraian
-- adierazten dena eginez lortzen den zerrenda berria itzuliko du:
--      * Zerrenda biak luzera berekoak ez badira errore mezua aurkeztu.
--      * Zerrenda biak hutsak badira zerrenda hutsa itzuli.
--      * Zerrenda bakoitzak elementu bakarra badu, bigarren zerrenda
--        dagoen bezalaxe itzuli.
--      * Lehenengo zerrendako posizioan True dagoen bakoitzean bigarren
--        zerrendan posizio horretan dagoen elementuak aurrera egingo du
--        hurrengo posiziokoarekin lekuz trukatuz eta zerrendak
--        zeharkatzen jarraituko da.
--      * Lehenengo zerrendako posizioan False dagoen bakoitzean bigarren
--        zerrendako elementua dagoen lekuan lagako da eta zerrendak
--        zeharkatzen jarraituko da.

-- Beraz, elementu batek posizio bat baino gehiago egingo ditu aurrera
-- True bat baino gehiago jarraian daudenean, True bakoitzeko posizio
-- bat hain zuzen ere. Baina elementu batek aurrera egin ahal izateko
-- atzerantz mugitzen den elementuak ez du inoiz aurrera egingo.
```

```
aurreratu:: [Bool] -> [Int] -> [Int]
aurreratu [] r
        | hutsa_da r            = []
        | otherwise             = error "Luzera desberdina"

aurreratu(x:s) r
        | (luzera (x:s)) /= (luzera r)          = error "Luzera desberdina"
        | hutsa_da s                            = r
        | x == True                             = (leh (hond r)):(aurreratu s ((leh(r)):(hond(hond r))))

        | otherwise                             = (leh r):(aurreratu s (hond r))

-------------------------------------------------------
--39
-- Zenbaki osoz osatutako zerrenda bat eta zerrenda horretako posizio bat
-- adierazten duen zenbaki oso bat emanda, posizio horretako elementua
-- bikoitia baldin bada, elementu hori kenduz gelditzen den zerrenda
-- itzultzen duen funtzioa.
-- Zehaztutako posizioko elementua bikoitia ez bada, ez da kendu behar.
-- Emandako zerrenda hutsa baldin bada edo zerrenda hutsa ez izanda posizio
-- bezala emandako zenbakia 1 eta zerrendaren luzeraren artean ez badago,
-- errore-mezua aurkeztuko da.

kendubikpos:: [Int] -> Int -> [Int]
kendubikpos [] pos = error "Zerrenda hutsa"


kendubikpos (x:s) pos
        | (pos < 1) || (pos > (luzera (x:s)))           = error "Ez da egokia"
        | (pos == 1) && (x `mod` 2 == 0)                = s
        | (pos == 1) && (x `mod` 2 /= 0)                = x:s
        | otherwise                                     = x:(kendubikpos s (pos - 1))

-------------------------------------------------------
--40
-- Int motako zerrenda bat emanda, jarraian dauden elementu berdinez osatutako
-- azpizerrenda bakoitzaren luzera bikoitia izan dadin, dagoeneko luzera
-- bikoitia duten azpizerrendak dauden bezala lagaz eta luzera bakoitia
-- duten azpizerrendei elementu berdin bat gehiago ipiniz osatzen den
-- zerrenda itzuliko duen funtzioa.
-- Emandako zerrenda hutsa bada, zerrenda hutsa itzuliko da.

azpiluzbikgehi:: [Int] -> [Int]
azpiluzbikgehi [] = []
```

```
azpiluzbikgehi (x:s)
        | hutsa_da s             = x:x:[]
        | x /= (leh s)           = x:x:(azpiluzbikgehi s)
        | otherwise              = x:x:(azpiluzbikgehi (hond s))

------------------------------------------------------------
--41
-- Zenbaki osoz osatutako zerrenda bat emanda, bikote bakoitzeko
-- elementu handiena bi aldiz ipiniz eta txikiena ezabatuz lortzen
-- den zerrenda itzultzen duen funtzioa. Bikoteak 1. eta 2. elementuaz,
-- 3. eta 4. elementuaz eta abar osatuta egongo dira.
-- Emandako zerrenda hutsa bada, zerrenda hutsa itzuliko da.
-- Emandako zerrendak luzera bakoitia badu, errore-mezua itzuliko da.

handibik:: [Int] -> [Int]
handibik [] = []

handibik (x:s)
        | luzera (x:s) `mod` 2 /= 0     = error "Luzera bakoitia"
        | x >= (leh s)                  = x:x:(handibik (hond s))
        | otherwise                     = (leh s):(leh s):(handibik (hond s))

------------------------------------------------------------
--42
-- Zenbaki osoz osatutako bi zerrenda emanda, lehenengo zerrendako elementu
-- denak edukitzeaz gain, lehenengo zerrendan 0 balioa agertzen den
-- bakoitzean bigarren zerrendako elementu bat (ezkerretik eskuinerako
-- ordena jarraituz) duen zerrenda itzultzen duen funtzioa.
-- Lehenengo zerrenda hutsa bada, zerrenda hutsa itzuliko da.
-- Lehenengo zerrenda hutsa ez denean, lehenengo zerrendako zero-kopurua,
-- bigarren zerrendako elementu-kopurua baino handiagoa baldin bada, errore-mezua
-- aurkeztuko da.

kokatu:: [Int] -> [Int] -> [Int]
kokatu [] r = []


kokatu (x:s) r
        | (aldiz 0 (x:s)) > (luzera r)          = error "Zero gehiegi"
        | x == 0                                = x:(leh r):(kokatu s (hond r))
        | otherwise                             = x:(kokatu s r)

------------------------------------------------------------
--43
-- Int motako zerrenda bat emanda, jarraian dauden elementu berdinez osatutako
-- azpizerrenda bakoitzaren luzera bikoitia izan dadin, dagoeneko luzera
-- bikoitia duten azpizerrendak dauden bezala lagaz eta luzera bakoitia
```

```
-- duten azpizerrendei elementu bat kenduz osatzen den zerrenda
-- itzultzen duen funtzioa.
-- Emandako zerrenda hutsa bada, zerrenda hutsa itzuliko da.

azpiluzbikken:: [Int] -> [Int]
azpiluzbikken [] = []


azpiluzbikken (x:s)
        | hutsa_da s            = []
        | x /= (leh s)          = azpiluzbikken s
        | otherwise             = x:x:(azpiluzbikken (hond s))

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