```
module Errek_zenb where


-- Oinarrizko errekurtsibitatea zenbakiekin


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

import Ez_errek

{- Ez_errek.hs moduluan dagoen "bikoitia" funtzioa erabili nahi delako
   "errusiar" izeneko funtzioa definitzeko.
-}

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

-- "bider" funtzioa: x balioa y aldiz batuz x * y kalkulatuko duen
-- funtzioa. Kasu berezi bezala, y balioa negatiboa baldin bada,
-- errore-mezua aurkeztuko du.

bider:: Int -> Int -> Int

bider x y
   | y < 0              = error "Bigarren balioa negatiboa da."
   | x == 0 || y == 0   = 0
   | x == 1             = y
   | otherwise          = x + bider x (y - 1)

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

-- "bneg" funtzioa: x * y balioa batuketaren bidez kalkulatuko duen
-- funtzioa. y balioa negatiboa denean ere kalkulua ondo egingo da.
-- Aurretik definitu den "bider" funtzioa erabiliko da laguntzaile
-- bezala. "bneg" funtzioa berez ez da errekurtsiboa, ez baitio
-- bere buruari deitzen. Errekurtsibitatea "bider" funtzioan dago.

bneg:: Int -> Int -> Int

bneg x y
   | x == 0 || y == 0          = 0
   | (x < 0) && (y < 0)        = bider (-x) (-y)
   | (x > 0) && (y > 0)        = bider x y
   | (x < 0) && (y > 0)        = bider x y
   | (x > 0) && (y < 0)        = bider (-x) (-y)

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

-- "errusiar" funtzioa: Errusiar biderketa bezala ezagutzen den
-- metodoari jarraituz x * y kalkulatuko duen funtzioa.
-- Kasu berezi bezala, y balioa negatiboa baldin bada, errore-mezua
-- aurkeztuko du.

errusiar:: Int -> Int -> Int

errusiar x y
   | y < 0              = error "Bigarren balioa negatiboa da."
   | y == 0             = 0
   | bikoitia y         = errusiar (x + x) (y `div` 2)
   | otherwise          = x + errusiar (x + x) (y `div` 2)

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

-- "zatios" funtzioa: y balioa x balioari zenbat aldiz kendu dakiokeen
-- zenbatuz, x eta y-ren arteko zatidura osoa kalkulatuko duen funtzioa.
-- Kasu berezi bezala, zatitzailea zero denean, errore-mezua aurkeztuko du.
-- Gainera, x edo y negatiboa baldin bada ere, errore-mezua aurkeztuko du.

zatios:: Int -> Int -> Int
```

```
zatios x y
   | y == 0             = error "Zatitzailea 0."
   | (x < 0) || (y < 0) = error "Gutxienez bietako bat negatiboa da."
   | x < y              = 0
   | otherwise          = 1 + zatios (x - y) y

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

-- "zatihond" funtzioa: y balioa baino txikiagoa den balio bat eduki
-- arte, x balioari eta ondoren lortuko diren kendurei y balioa kenduz,
-- x eta y-ren arteko zatidura osoaren hondarra kalkulatuko duen funtzioa.
-- Kasu berezi bezala, zatitzailea zero denean, errore-mezua aurkeztuko du.
-- Gainera, x edo y negatiboa baldin bada ere, errore-mezua aurkeztuko du.

zatihond:: Int -> Int -> Int

zatihond x y
   | y == 0             = error "Zatitzailea 0."
   | (x < 0) || (y < 0) = error "Gutxienez bietako bat negatiboa da."
   | x < y              = x
   | otherwise          = zatihond (x - y) y

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