7. GAIKO ARIKETAK
HASKELL

A) Zenbaki osoen gaineko eragiketak

Jarraian aipatzen diren eragiketak definitzen dituzten ekuazioak eman. Eragiketak Int datu-motarentzat dira:

1) faktoriala

Zero edo handiagoa den zenbaki oso bat emanda, bere faktoriala itzultzen duen funtzioa. Emandako zenbakia negatiboa baldin bada, errore-mezua aurkeztu behar du: *faktoriala*.

0 zenbakiaren faktoriala 1 da. 0 baino handiagoa den n zenbaki baten faktoarial n – 1 zenbakiaren faktoariala bider n da.

Adibideak:

faktoriala 2 = 2
faktoriala 3 = 6
faktoriala 4 = 24

2) fib

Zero edo handiagoa den zenbaki oso bat emanda, bere fibonaccia itzultzen duen funtzioa. Emandako zenbakia negatiboa baldin bada, errore-mezua aurkeztu behar du: *fib*.

0 zenbakiaren fibonaccia 0 da, 1 zenbakiaren fibonaccia 1 da, 2 baino handiagoa edo berdina den n zenbaki baten fibonaccia n – 1 eta n – 2 zenbakien fibonaccien batura da.

Adibideak:

fib 2 = fib 0 + fib 1 = 0 + 1 = 1
fib 3 = fib 1 + fib 2 = 1 + 1 = 2
fib 4 = fib 2 + fib 3 = 1 + 2 = 3
fib 5 = fib 3 + fib 4 = 2 + 3 = 5
fib 6 = fib 4 + fib 5 = 3 + 5 = 8
fib 7 = fib 5 + fib 6 = 5 + 8 = 13
fib 8 = fib 6 + fib 7 = 8 + 13 = 21

B) Zerrenden gaineko eragiketak

Jarraian aipatzen diren eragiketak definitzen dituzten ekuazioak eman. Eragiketak [Int] eta [t] datu-motentzat dira:

1) inkr

Osoak diren zenbakiz osatutako zerrenda bat emanda, zerrendako elementu bakoitzari 1 gehituz (inkrementatuz) lortzen den zerrenda itzultzen duen funtzioa: *inkr*.

Adibideak:

inkr [4, 8, 5] = [5, 9, 6]
inkr [] = []

2) batu

Osoak diren zenbakiz osatutako zerrenda bat emanda, zerrendako elementu denen batura itzultzen duen funtzioa: *batu*.

Adibideak:

batu [4, 8, 5] = 17
batu [] = 0

3) bikoitirik

Osoak diren zenbakiz osatutako zerrenda bat emanda, zerrendan zenbaki bikoitirik baldin badago True eta bestela False itzultzen duen funtzioa: *bikoitirik*.

Adibideak:

bikoitirik [4, 8, 5] = True
bikoitirik [11, 7, 5, 21] = False
bikoitirik [] = False

Aurretik definituta dauden beste funtzio hauek erabili: **bikoitia**, **bakoitia**

4) bik_posi

Osoak diren zenbakiz osatutako bi zerrenda emanda, zerrenda bateko posizio batean zenbaki bikoiti bat dagoen bakoitzean beste zerrendako posizio berean ere zenbaki bikoiti bat al dagoen erabakitzen duen funtzioa. Zerrenda biak luzera berekoak ez badira, errore-mezua aurkeztu behar du: *bik_posi*.

Adibideak:
bik_posi [4, 8, 3] [2, 8, 11] = True
bik_posi [4, 8, 3] [9, 2, 17] = False
bik_posi [5, 7] [1, 9, 3] = error "luzera ezberdina"
bik_posi [5, 7] [] = error "luzera ezberdina"

Aurretik definituta dauden beste funtzio hauek erabili: **luzera**, **hutsa_da, leh, hond, bikoitia**, **bakoitia**

5) azkena

Zerrenda bat emanda, azkeneko elementua (eskuineko ertzean dagoena) itzultzen duen funtzioa. Zerrenda hutsa bada, errore-mezua aurkeztu behar du: *azkena*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
azkena [d4, d8, d3, d5, d2] = d2
azkena [] = error "Zerrenda hutsa."
```

Aurretik definituta dagoen beste funtzio hau erabili: **hutsa_da**

6) azkena_kendu

Zerrenda bat emanda, azkeneko elementua (eskuineko ertzean dagoena) kenduz gelditzen den zerrenda itzultzen duen funtzioa. Zerrenda hutsa bada, errore-mezua aurkeztu behar du: *azkena_kendu*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
azkena_kendu [d4, d8, d3, d5, d2] = [d4, d8, d3, d5]
azkena_kendu [] = error "Zerrenda hutsa."
```

Aurretik definituta dagoen beste funtzio hau erabili: **hutsa_da**

7) alder

Zerrenda bat emanda, bere alderantzizkoa itzultzen duen funtzioa: *alder*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
alder [d4, d8, d3, d5, d2] = [d2, d5, d3, d8, d4]
alder [] = []
```

Bi eratara egin:

a) Aurretik definituta dagoen beste funtzio hau erabiliz: ++

b) Aurretik definituta dauden beste bi funtzio hauek erabiliz: **azkena**, **azkena_kendu**

8) alder_dira

Bi zerrenda emanda, alderantzizkoak al diren ala ez erabakitzen duen funtzioa: *alder_dira*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
alder_dira [d4, d8, d3] [d3, d8, d4] = True
alder_dira [] [] = True
alder_dira [d1, d8, d6] [d3, d8] = False
alder_dira [] [d3, d8] = False
```

Bi eratara egin:

a) Aurretik definituta dagoen beste funtzio hau erabiliz: **alder**

b) Aurretik definituta dauden beste funtzio hauek erabiliz: **azkena**, **azkena_kendu**, **luzera** eta **hutsa_da**

9) aldiz

Elementu bat eta zerrenda bat emanda, elementu hori zerrendan zenbat aldiz agertzen den kontatzen duen funtzioa: *aldiz*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
aldiz d8 [d3, d7, d8, d20, d8, d7] = 2
aldiz d10 [d3, d7, d8, d20, d8, d7] = 0
aldiz d6 [] = 0
```

10) erre

Zerrenda batean errepikatutako elementurik ba al dagoen erabakitzen duen funtzioa: *erre*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
erre [d5, d6, d20, d6, d6, d10, d34] = True
erre [d5, d6, d20, d6, d6, d5, d34] = True
erre [d5, d6, d20, d8, d7] = False
erre [] = False
```

Aurretik definituta dagoen beste funtzio hau erabili: **badago**

11) kendu

Elementu bat eta zerrenda bat emanda, elementu horren agerpen denak kenduz gelditzen den zerrenda itzultzen duen funtzioa: *kendu*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
kendu d6 [d5, d6, d20, d6, d6, d10, d34] = [d5, d20, d10, d34]
kendu d6 [d5, d20, d34] = [d5, d20, d34]
kendu d2 [] = []
```

12) bik_kendu

Zenbaki osoz osatutako zerrenda bat emanda, elementu bikoiti denak kenduz lortzen den zerrenda itzultzen duen funtzioa: *bik_kendu*.

Adibideak:

```
bik_kendu [5, 6, 20, 11, 7, 10, 34] = [5, 11, 7]
bik_kendu [4, 20, 18] = []
bik_kendu [] = []
```

Aurretik definituta dauden beste bi funtzio hauek erabili: **bikoitia**, **bakoitia**

13) pos_bik_kendu

Zerrenda bat emanda, posizio bikoitietan dauden elementuak kenduz lortzen den zerrenda itzultzen duen funtzioa: *pos_bik_kendu*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
pos_bik_kendu [d5, d6, d20, d7, d6, d10, d34] = [d5, d20, d6, d34]
pos_bik_kendu [d5, d20, d34] = [d5, d34]
pos_bik_kendu [] = []
```

Aurretik definituta dauden beste funtzio hauek erabili: **hutsa_da, hond**

14) pos_bak_kendu

Zerrenda bat emanda, posizio bakoitietan dauden elementuak kenduz lortzen den zerrenda itzultzen duen funtzioa: *pos_bak_kendu*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
pos_bak_kendu [d5, d6, d20, d7, d6, d10, d34] = [d6, d7, d10]
pos_bak_kendu [d5, d20, d34] = [d20]
pos_bak_kendu [] = []
```

Aurretik definituta dauden beste funtzio hauek erabili: **hutsa_da**, **leh, hond**

15) erre_kendu

Zerrenda bat emanda, errepikapenak kenduz gelditzen den zerrenda itzultzen duen funtzioa. Elementu bakoitzaren kasuan bere lehenengo agerpena mantendu behar da: *erre_kendu*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
erre_kendu [d5, d6, d20, d6, d6, d10] = [d5, d6, d20, d10]
erre_kendu [d4, d4, d8, d22, d8, d22] = [d4, d8, d22]
erre_kendu [] = []
```

Aurretik definituta dauden beste bi funtzio hauek erabili: **badago**, **kendu**

16) erre_kendu2

Zerrenda bat emanda, errepikapenak kenduz gelditzen den zerrenda itzultzen duen funtzioa. Elementu bakoitzaren kasuan bere azkenengo agerpena mantendu behar da: *erre_kendu2*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
erre_kendu2 [d5, d6, d20, d6, d6, d10] = [d5, d20, d6, d10]
erre_kendu2 [d4, d4, d8, d22, d8, d5] = [d4, d22, d8, d5]
erre_kendu2 [] = []
```

Aurretik definituta dagoen beste funtzio hau erabili: **badago**

17) bikote_berdin

Zerrenda bat emanda, berdinak diren bi elementu jarraian agertzen al diren ala ez erabakitzen duen funtzioa: *bikote_berdin*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
bikote_berdin [d5, d9, d3, d3, d2] = True
bikote_berdin [d5, d3, d3, d3, d2] = True
bikote_berdin [d5, d5, d1, d2, d2] = True
bikote_berdin [d5, d9, d3, d2, d3] = False
bikote_berdin [] = False
```

Aurretik definituta dauden beste bi funtzio hauek erabili: **hutsa_da**, **leh**

18) aurrizkia

Zerrenda bat beste zerrenda baten aurrizkia al den erabakitzen duen funtzioa: *aurrizkia*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
aurrizkia [d5, d9] [d5, d9, d3, d3, d2] = True
aurrizkia [] [d5, d9, d3, d3, d2] = True
aurrizkia [d7, d5, d10] [d5, d9, d3, d3, d2] = False
aurrizkia [d5, d9, d8, d4] [d5, d9] = False
aurrizkia [d5, d6] [] = False
```

Aurretik definituta dauden beste funtzio hauek erabili: **hutsa_da**, **leh, hond**

Orokorrean s zerrenda bat r zerrenda baten aurrizkia izango da s ++ w = r betetzen duen w zerrenda existitzen bada (horregatik r zerrenda bat hartuta, zerrenda hutsa bere aurrizkia izango da [] ++ r = r betetzen delako)

19) azpizer

Zerrenda bat beste zerrenda baten azpizerrenda al den erabakitzen duen funtzioa: *azpizer*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

```
azpizer [d9, d3] [d5, d9, d3, d3, d2] = True
azpizer [d5, d9, d3] [d5, d9, d3, d3, d2] = True
azpizer [] [d5, d9, d3, d3, d2] = True
azpizer [d2, d5, d3] [d5, d9, d3, d3, d2] = False
azpizer [d5, d9, d8, d4] [d5, d9] = False
azpizer [d5, d6] [] = False
```

Orokorrean s zerrenda bat r zerrenda baten azpizerrenda izango da v ++ s ++ w = r betetzen duten v eta w bi zerrenda existitzen badira. Horregatik r edozein zerrenda hartuta, zerrenda hutsa bere azpizerrenda izango da [] ++ [] ++ r = r betetzen delako (Hor v zerrenda [] izango da eta w zerrenda r izango da).

Aurretik definituta dauden beste funtzio hauek erabili: **aurrizkia, hutsa_da, hond**

20) elem_pos

Posizio bat (zenbaki oso bat) eta zerrenda bat emanda, zerrendako posizio horretan dagoen elementua itzultzen duen funtzioa. Emandako posizioa tarte egokian ez badago (1 eta zerrendaren luzeraren artean), errore-mezu bat aurkeztu behar da: *elem_pos*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

elem_pos 2 [d5, **d9**, d3, d3, d2] = d9
elem_pos 8 [d5, d9, d3, d3, d2] = error "Ez da egokia"
elem_pos 0 [d5, d9, d3, d3, d2] = error "Ez da egokia"
elem_pos 2 [] = error "Ez da egokia"

Aurretik definituta dagoen beste funtzio hau erabili: **luzera**

21) sartu

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 gelditu behar dute. Emandako posizioa tarte egokian ez badago (1 eta zerrendaren luzera gehi 1en artean), errore-mezu bat aurkeztu behar da: *sartu*.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira)

sartu 2 d8 [d5, **d9**, d3, d3, d2] = [d5, **d8**, d9, d3, d3, d2]
sartu 6 d8 [d5, d9, d3, d3, d2] = [d5, d9, d3, d3, d2, **d8**]
sartu 1 d8 [d5, d9, d3, d3, d2] = [**d8**, d5, d9, d3, d3, d2]
sartu 0 d8 [d5, d9, d3, d3, d2] = error "Ez da egokia"
sartu 9 d8 [d5, d9, d3, d3, d2] = error "Ez da egokia"
sartu 1 d8 [] = [d8]
sartu 2 d8 [] = error "Ez da egokia"

Aurretik definituta dagoen beste funtzio hau erabili: **luzera**

22) lehen_bik_pos

Zenbaki osoz osatutako zerrenda bat emanda, zerrendako lehenengo zenbaki bikoitiaren posizioa itzultzen duen funtzioa. Zenbaki bikoitirik ez badago, zerrendaren luzera gehi 1 itzuli behar du: *lehen_bik_pos*.

Adibideak:

lehen_bik_pos [5, 9, 8, 7, 6] = 3
lehen_bik_pos [9, 5, 5, 3, 79] = 6

23) azken_pos

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 itzuli behar da: *azken_pos*.

Adibideak:

azken_pos 8 [8, 9, 8, 17, 6] = 3
azken_pos 8 [7, 3, 0, 0, 97] = 0

Aurretik definituta dagoen beste funtzio hau erabili: **badago**

24) hand

Zenbaki osoz osatutako zerrenda bat emanda, zerrendako balio handiena itzultzen duen funtzioa. Zerrenda hutsa bada, errore-mezua aurkeztu behar da: *hand*.

Adibidea:

hand [5, 3, 8, 7, 8] = 8

Aurretik definituta dauden beste funtzio hauek erabili: **hutsa_da, leh, hond**

25) elkartu

t motako bi zerrenda emanda, lehenengo zerrendako lehenengo elementua eta bigarren zerrendako lehenengo elementua ondoan ipiniz zerrendak elkartuz lortzen den zerrenda itzultzen duen funtzioa.

Adibideak: (*di* bezala agertzen direnak t motako elementuak dira):

elkartu [d2, d5, d4, d3] [**d8, d9**] = [d3, d4, d5, d2, **d8, d9**]

Bigarren zerrendako elementuak orden berean gelditzuko dira baina lehenengo zerrendako elementuak alderantzizko ordenean gelditzuko dira.

Bi eratara egin:

a) Aurretik definitutako funtziorik erabili gabe.
b) Aurretik definituta dauden beste funtzio hauek erabiliz: **alder** eta ++

26) ezabatu

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

Adibidea:

ezabatu 2 [5, 3, 8, 7, 8] = [8, 7, 8]

Aurretik definituta dagoen beste funtzio hau erabili: **luzera**

27) hartu

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 hartu beharko dira. Zenbakia 0 eta zerrendaren luzeraren artean ez badago, errore-mezua aurkeztu behar da. Zenbakia 0 bada, zerrenda hutsa itzuliko da.

Adibidea:

hartu 2 [5, 3, 8, 7, 8] = [5, 3]

Aurretik definituta dagoen beste funtzio hau erabili: **luzera**

28) kolapsatu

Zenbaki osoz osatutako zerrenda bat emanda, jarraian errepikatuta agertzen diren elementuen kopia bakarra mantenduz osatzen den zerrenda itzultzen duen *kolapsatu* izeneko funtzioa. Beraz funtzioak jarraian dauden kopiak ezabatu behar ditu, kopia bakarra lagaz. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da. (*hutsa_da* eta *leh* funtzioak erabili):

1. adibidea:

kolapsatu [0, 8, 8, 8, 4, 0, 0] = [0, 8, 4, 0]

2. adibidea:

kolapsatu [3, 9, 9, 3] = [3, 9, 3]

29) hedatu

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. (*hutsa_da*, *leh* eta *hond* funtzioak erabili):

Adibidea:

hedatu [4, 5, 2, 10, 8, 3, 6] =
= [4, 5 + 4, 2, 10 + 2, 8 + 12, 3 + 20, 6]
= [4, 9, 2, 12, 20, 23, 6]

4 bikoitia denez hedatu egingo da
9 bakoitia denez ez da hedatuko
2 hedatu egingo da, bikoitia baita
12 ere hedatu egingo da bikoitia delako
20 ere bai
23 ez, bakoitia baita
6 ere ez da hedatuko, zerrendako azkeneko posizioan dago eta

30) zapaldu

Zenbaki osoz osatutako zerrenda bat emanda, ezkerretik hasi eta elementu bakoitzak bere atzetik datozen zenbaki txikiagoak (handiagoa edo berdina den bat agertu arte) zapalduz (ordezkatuz) lortzen den zerrenda itzultzen duen funtzioa eman. 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 (*hutsa_da*, *leh* eta *hond* funtzioak erabili):

1. adibidea:

zapaldu [4, 2, 3, 2, 8, 4, 9] = [4, 4, 4, 4, 8, 8, 9]

2. adibidea:

zapaldu [3, 9, 9, 3] = [3, 9, 9,9]

31) hondoratu

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 funtzioa eman. Kasu honetan lehenengo elementua baino handiagoa edo berdina den zenbakia aurkitzen denean prozesua bukatu egingo da, eta ondoren dauden elementuak berdin gelditzuko dira. Lehenengo elementua beste denak baino handiagoa baldin bada, bukaerako posizioan (eskuineko ertzean) geratuko da. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da eta hasierako zerrendak elementu bakarra badu, zerrenda hori bera itzuliko da (*hutsa_da*, *leh* eta *hond* funtzioak erabili):

1. adibidea:

hondoratu [**5**, 2, 4, 1, 8, 4, 3] = [2, 4, 1, **5**, 8, 4, 3]

Zerrendako lehenengo elementua (5) bera baino handiagoa edo berdina den zenbaki bat (8) aurkitu arte hondoratu (edo eskuinera desplazatu) da.

2. adibidea:

hondoratu [**8**, 3, 2, 9] = [3, 2, **8**, 9]

Zerrendako lehenengo elementua (8) bera baino handiagoa edo berdina den zenbaki bat (9) aurkitu arte hondoratu (edo eskuinera desplazatu) da.

3. adibidea:

hondoratu [**8**, 9, 1, 4] = [**8**, 9, 1, 4]

Zerrendako lehenengo elementua (8) ezin izan da desplazatu bigarrena handiagoa delako.

4. adibidea:

hondoratu [**8**, 5, 1, 4] = [5, 1, 4, **8**]

Lehenengo elementua (8) beste denak baino handiagoa denez, eskuineko ertzean geratu da.

32) elem_kendu

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 *elem_kendu* izeneko funtzioa eman. Zerrenda hutsa bada, errore-mezu bat aurkeztu behar da eta zerrenda hutsa ez bada baina posizioa ez badago 1 eta zerrendaren luzeraren artean, orduan ere errore-mezu bat aurkeztu behar da. (*luzera* funtzioa erabili):

1. adibidea:

elem_kendu [5, 4, 1, 2, **4**, 8, 3] 3 = [5, 4, 1, 2, 8, 3]

2. adibidea:

elem_kendu [1, 7, **5**, 8] 2 = [1, 7, 8]

33) garbitu

Zenbaki osoz osatutako zerrenda bat emanda, zerrendako lehenengo elementua baino txikiagoak diren elementu denak kenduz eta lehenengo elementu hori bukaeran ipiniz lortzen den zerrenda itzultzen duen garbitu izeneko funtzioa eman. Emandako zerrenda hutsa bada, errore-mezua itzuli beharko du funtzioak. Aurretik definituta dauden *hutsa_da* (zerrenda bat hutsa al den erabakitzen duen funtzioa), *leh* (zerrendako lehenengo elementua itzultzen duen funtzioa) eta *hond* (zerrendako lehenengo elementua kenduz gelditzen den zerrenda itzultzen duen funtzioa) erabili:

Adibideak:
garbitu [5, 8, 8, 4, 9, 3] = [8, 8, 9, 5]
garbitu [5, 8, 5, 4, 9, 3] = [8, 5, 9, 5]
garbitu [5, 8, 8, 9, 6] = [8, 8, 9, 6, 5]

Laburtuz, lehenengo elementuak zerrenda osoa zeharkatu beharko du bera baino txikiagoak direnak desageraraziz eta bera bukaeran geldituz.

34) zeharkatu

Zenbaki osoz osatutako zerrenda bat emanda, zerrendako lehenengo elementua behean zehazten dena bete arte eskuinera desplazatuz lortzen den zerrenda itzultzen duen *zeharkatu* izeneko funtzioa eman:

Noiz arte desplazatu:

- Lehenengo elementu hori zerrenda osoan ez bada agertzen, bukaerara iritsi arte desplazatu.
- Lehenengo elementu hori zerrendan agertzen bada, zenbakia bera agertu arte desplazatu eta zenbakiaren kopia biak bi zeroz ordezkatu.

Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli.

Ekuazioak ematerakoan **hutsa_da**, **leh** eta **hond** erabili:

Adibideak:
zeharkatu [**5**, 8, 4, **5**, 9, 3] = [8, 4, **0**, **0**, 9, 3]
zeharkatu [**5**, 8, 8, 4, 9, 3] = [8, 8, 4, 9, 3, **5**]
zeharkatu [**5**, **5**, 5, 5] = [0, 0, **5**, **5**]

Laburtuz, 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.

35) zatitu

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 *zatitu* izeneko funtzioa eman:

- 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 aurkeztu behar da eta hasierako zerrendak elementu bakarra badu, zerrenda hutsa itzuli beharko da. Zerrenda hutsa bada True eta bestela False itzultzen duen **hutsa_da** eta zerrendako lehenengo elementua eta lehenengo elementua kenduz gelditzen den zerrenda itzultzen dituzten **leh** eta **hond** funtzioak erabili:

Adibideak:

zatitu [**2**, 8, 6, 5, 9, 20] = [4, 3, 5, 9, 10] zatitu [**5**, 8, 8, 15, 9] = [8, 8, 3, 9]	zatitu [**3**, 15] = [5] zatitu [**3**, 14] = [14]

Laburtuz, lehenengo elementuak zerrenda osoa zeharkatu beharko du zatitu ditzakeen elementuak zatituz eta azkenean bera desagertu egingo da.

36) superbik

Zenbaki osoz osatutako zerrenda bat emanda, zerrendako zenbaki bikoiti denen batura bukaeran ipiniz eta zenbaki bakoitiak mantenduz lortzen den zerrenda itzultzen duen *superbik* izeneko funtzioa eman.

Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli beharko da. Hasierako zerrendak elementu bakarra badu, zerrenda bera itzuli beharko da.

Zerrenda hutsa bada True eta bestela False itzultzen duen **hutsa_da** eta zerrendako lehenengo elementua eta lehenengo elementua kenduz gelditzen den zerrenda itzultzen dituzten **leh** eta **hond** funtzioak erabili:

Adibideak:

superbik [**2**, **8**, 5, **14**, 9, **10**] = [5, 9, **34**]	superbik [5, 3, 7, 1] = [5, 3, 7, 1]

Ezkerretik hasiz, zerrenda zeharkatu beharko da. Zenbaki bakoitiak mantendu egingo dira, ez da ezer egin behar beraiekin. Zenbaki bikoiti bat aurkitzean, hurrengo posizioan zenbaki bakoiti bat badago, bikoitia eskuinera desplazatuko da (bikoitia eta bakoitia lekuz trukatuko dira). Baina zenbaki bikoiti bat aurkitzean, hurrengo posizioan zenbaki bikoiti bat badago, zenbaki bi horiek desagertu egingo dira eta beraien ordez bien batura gelditzuko da.

37) metatu

Zenbaki osoz osatutako zerrenda bat emanda, posizio bakoitzean sarrerako zerrendako lehenengo posiziotik posizio horretara arteko batura duen zerrenda itzultzen duen *metatu* izeneko funtzioa eman. Emandako zerrenda hutsa bada, zerrenda hutsa itzuli beharko du funtzioak. Aurretik definituta dauden *hutsa_da* (zerrenda bat hutsa al den erabakitzen duen funtzioa), *leh* (zerrendako lehenengo elementua itzultzen duen funtzioa) eta *hond* (zerrendako lehenengo elementua kenduz gelditzen den zerrenda itzultzen duen funtzioa) erabili.

Adibideak: metatu [10, 8, 15] = [10, 18, 33]
metatu [10, 0, 8] = [10, 10, 18]

Zerrenda zeharkatuz eta posizio bakoitzera arte metatutako batura kalkulatuz joan beharko da.

38) aurreratu

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 itzultzen duen "aurreratu" izeneko funtzioaren **espezifikazio ekuazionala** eman:

- 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.

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.

Aurretik definituta dauden *luzera* (zerrenda bateko elementu-kopurua kalkulatzen duen funtzioa), *hutsa_da* (zerrenda bat hutsa al den erabakitzen duen funtzioa), *leh* (zerrendako lehenengo elementua itzultzen duen funtzioa) eta *hond* (zerrendako lehenengo elementua kenduz gelditzen den zerrenda itzultzen duen funtzioa) erabili.

Adibideak:

- aurreratu [False, **True**, **True**, False, **True**] [8, 3, 9, 5, 2] = [8, 9, 5, 3, 2]
  Kasu honetan 8 lehenengo posizioan geldituko da False dagoelako. Bigarren posizioan True dagoenez 3a posizio bat aurreratuko da 9arekin leku-aldaketa eginez. 3a hirugarren posizioan kokatu ondoren, hirugarren posizioan ere True dagoenez posizio bat aurreratuko da 5arekin leku-aldaketa eginez. Laugarren posizioan False dagoenez, 3ak ez du jarraituko aurrera. Bosgarren posizioan True egon arren, 2 ez da mugituko ez baitu eskuinerantz egiteko lekurik.

- aurreratu [**True**, **True**, **True**, False, False] [8, 3, 9, 5, 2] = [3, 9, 5, 8, 2]
  Kasu honetan lehenengo posizioan True dagoenez 8a posizio bat aurreratuko da 3arekin leku-aldaketa eginez. Bigarren posizioan kokatu ondoren, berriro ere True dagoenez 8ak posizio bat aurrera egingo du 9arekin leku-trukaketa eginez. 8a hirugarren posizioan kokatu ondoren, hirugarren posizioan berriro ere True dagoenez 8ak posizio bat aurrera egingo du 5arekin leku-aldaketa eginez. Laugarren posizioan False

dagoenez, 8ak ez du jarraituko aurrera. 2 ez da mugituko ez baitu eskuinerantz egiteko lekurik eta gainera False dago bosgarren posizioan.

- aurreratu [**True**, False, **True**, False, **True**] [8, 3, 9, 5, 2] = [3, 8, 5, 9, 2]
- aurreratu [**True**, False, **True**, **True**, **True**] [8, 3, 9, 5, 2] = [3, 8, 5, 2, 9]

39) kendubikpos

a) 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 *kendubikpos* izeneko funtzioa eman. 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 aurkeztu beharko da.

Ekuazioak ematerakoan *luzera* izeneko funtzioa erabili:

Adibideak:

- kendubikpos [8, 5, **9**, 7, 10, 4] **3** = [8, 5, **9**, 7, 10, 4]

  3 posizioko elementua bikoitia ez denez ez da kendu behar eta horregatik hasierako zerrenda itzuli da kasu honetan.

- kendubikpos [8, 5, **16**, 7, 10, 4] **3** = [8, 5, 7, 10, 4]

  3 posizioko elementua bikoitia denez, posizio horretako elementua kenduz lortzen den zerrenda itzuli da kasu honetan.

40) azpiluzbikgehi

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 itzultzen duen *azpiluzbikgehi* izeneko funtzioa eman. Emandako zerrenda hutsa bada, zerrenda hutsa itzuli beharko du funtzioak. Aurretik definituta dauden *hutsa_da* (zerrenda bat hutsa al den erabakitzen duen funtzioa), *leh* (zerrendako lehenengo elementua itzultzen duen funtzioa) eta *hond* (zerrendako lehenengo elementua kenduz gelditzen den zerrenda itzultzen duen funtzioa) erabili:

Adibideak:
azpiluzbikgehi [10, 10, 10, 8, 8, 15, 8] = [10, 10, 10, **10**, 8, 8, 15, **15**, 8, **8**]
azpiluzbikgehi [15] = [15, 15]
azpiluzbikgehi [15, 10, 15, 10] = [15, **15**, 10, **10**, 15, **15**, 10, **10**]

- Lehenengo adibidean elementu berdinez osatutako lau azpizerrenda daude. Lehenengo azpizerrendan 10 balioa hiru aldiz agertzen da eta kopurua bikoitia izan dadin laugarren 10a gehitu da. Hirugarren eta laugarren azpizerrendetan ere elementu bat gehitu da luzera bikoitiko azpizerrendak edukitzeko.
- Bigarren adibidean azpizerrenda bakarra dago (15a) eta gainera elementu kopurua bakoitia denez, beste 15 bat gehitu da.
- Hirugarren adibidean elementu berdinez osatutako lau azpizerrenda daude. Azpirrenda bakoitza elementu batez osatuta dago eta kasu bakoitzean kopurua bikoitia izan dadin beste elementu bat gehitu da.

41) handibik

Zenbaki osoz osatutako zerrenda bat emanda, bikote bakoitzeko elementu handiena bi aldiz ipiniz eta txikiena ezabatuz lortzen den zerrenda itzultzen duen *handibik* izeneko funtzioa eman. Emandako zerrenda hutsa bada, zerrenda hutsa itzuli beharko du funtzioak. Emandako zerrendak luzera bakoitia badu, errore-mezua itzuli beharko du funtzioak. Aurretik definituta dauden *luzera* (zerrenda baten luzera edo elementu-kopurua itzultzen duen funtzioa), *leh* (zerrendako lehenengo elementua itzultzen duen funtzioa) eta *hond* (zerrendako lehenengo elementua kenduz gelditzen den zerrenda itzultzen duen funtzioa) erabili:

Adibideak:

handibik [10, 8, 5, 7, 7, 20, 5, 5] = [10, 10, 7, 7, 20, 20, 5, 5]

Adibideko zerrendan lau bikote daude. Lehenengoan 10 denez handiena, zerrenda berriko lehenengo bikotea bi 10ez osatuta dago. Bigarren bikotean 7 da haundiena eta horregatik zerrenda berriko bigarren bikotean bi 7 ditugu. Hirugarren bikotean 20 da handiena eta zerrenda berriko hirugarren bikotean bi 20 ditugu. Laugarren bikotean 5 balioa da handiena eta zerrenda berriko laugarren bikotean bi 5 ditugu.

42) kokatu

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 *kokatu* izeneko funtzioa eman.

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.

Erabili beharreko funtzioak:

- Elementu bat eta zerrenda bat emanda, elementua zerrendan zenbat aldiz agertzen den itzultzen duen *aldiz* funtzioa.
- Zerrenda bat emanda, zerrendako elementu-kopurua itzultzen duen *luzera* izeneko funtzioa.
- Zerrenda bat emanda, zerrendako lehenengo elementua itzultzen duen *leh* izeneko funtzioa.
- Zerrenda bat emanda, zerrendako lehenengo elementua kenduz lortzen den zerrenda itzultzen duen *hond* izeneko funtzioa.

Adibideak:

- kokatu [8, **0**, **0**, 7, **0**, 6] [**3**, **20**, **12**, 45, 28] = [8, **0**, **3**, **0**, **20**, 7, **0**, **12**, 6]

  Lehenengo zerrendan hiru zero daudenez, zerrenda berrian bigarren zerrendako lehenengo hiru elementuak sartu dira, bakoitza zero baten ondoren kokatuz.

- kokatu [8, **0**, **0**, 7, **0**, 6] [**3**, **20**]

  Kasu honetan funtzioak errore-mezua aurkeztu beharko luke bigarren zerrendako elementu-kopurua ez delako nahikoa, lehenengo zerrendan hiru zero daude eta bigarrengoan bakarrik bi elementu daude.

43) azpiluzbikken

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 *azpiluzbikken* izeneko funtzioa eman. Emandako zerrenda hutsa bada, zerrenda hutsa itzuli beharko du funtzioak. Aurretik definituta dauden *hutsa_da* (zerrenda bat hutsa al den erabakitzen duen funtzioa), *leh* (zerrendako lehenengo elementua itzultzen duen funtzioa) eta *hond* (zerrendako lehenengo elementua kenduz gelditzen den zerrenda itzultzen duen funtzioa) erabili:

Adibideak:

azpiluzbikken [10, 10, **10**, 8, 8, **15**, **8**] = [10, 10, 8, 8]
azpiluzbikken [**15**] = []
azpiluzbikken [**15**, **10**, **15**, **10**] = []
azpiluzbikken [10, 10, 10, 10, 8, 8] = [10, 10, 10, 10, 8, 8]

- Lehenengo adibidean elementu berdinez osatutako lau azpizerrenda daude. Lehenengo azpizerrendan 10 balioa hiru aldiz agertzen da eta kopurua bikoitia izan dadin hirugarren 10a kendu egin da. Hirugarren eta laugarren azpizerrendetan ere elementu bat kendu da eta azpizerrenda horiek elementu bakarrekoak zirenez beraien aztarnarik ez da gelditu.
- Bigarren adibidean azpizerrenda bakarra dago (15a duena) eta gainera elementu-kopurua bakoitia denez, elementu bat kendu egin behar izan da, zerrenda hutsa geldituz.
- Hirugarren adibidean elementu berdinez osatutako lau azpizerrenda daude. Azpirrenda bakoitza elementu bakar batez osatuta dago eta kasu bakoitzean kopurua bikoitia izan dadin elementu bat kendu da eta ondorioz zerrenda hutsa gelditu da.
- Laugarren adibidean elementu berdinez osatutako bi azpizerrenda daude. Azpirrenda bakoitzean elementu-kopurua bikoitia denez ez da ezer kendu behar eta hasierako zerrenda bera geratu da azkenean.

C) Murgilketa

1) faktoreak

2 baino handiagoa edo berdina den x zenbakiaren **errepikapenik gabe**ko faktore lehenen zerrenda kalkulatzen duen funtzioa definitu murgilketaren teknika erabiliz. Zenbaki bat lehena al den erabakitzeko, aurretik definituta dagoen *lehena* izeneko funtzioa erabili behar da. Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:

faktoreak 8 = [2]	faktoreak 12 = [2, 3]
faktoreak 15 = [3, 5]	faktoreak 7 = [7]
faktoreak 100 = [2, 5]	faktoreak 75 = [3, 5]

Helburua *faktoreak* funtzioa definitzeko *faktoreak_lag* beste funtzio bat definitzea da. *faktoreak_lag* funtzioak, x eta bm bi zenbaki oso emanda, x zenbakiaren [bm..x] tarteko faktore lehenen zerrenda itzuli beharko du (**errepikapenik gabe**). Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da eta bm balioa 2 baino txikiagoa baldin bada ere, errore-mezua aurkeztu beharko da.

Adibideak:

faktoreak_lag 8 2= [2]	faktoreak_lag 8 4= []
faktoreak_lag 12 3= [3]	faktoreak_lag 15 4= [5]
faktoreak_lag 7 3 = [7]	faktoreak_lag 7 10 = []
faktoreak_lag 100 3 = [5]	faktoreak_lag 75 4 = [5]

2) desk

2 baino handiagoa edo berdina den x zenbakiaren **errepikapendun** faktore lehenen zerrenda kalkulatzen duen funtzioa definitu murgilketaren teknika erabiliz. Zenbaki bat lehena al den erabakitzeko, aurretik definituta dagoen *lehena* izeneko funtzioa erabili behar da. Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:

desk 8 = [2, 2, 2]	desk 12 = [2, 2, 3]	desk 100 = [2, 2, 5, 5]
desk 15 = [3, 5]	desk 7 = [7]	desk 75 = [3, 5, 5]

Helburua *desk* funtzioa definitzeko *desk_lag* beste funtzio bat definitzea da. *desk_lag* funtzioak, x eta bm bi zenbaki oso emanda, x zenbakiaren [bm..x] tarteko faktore lehenen zerrenda itzuli beharko du (**errepikapenekin**). Emandako x balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da eta bm balioa 2 baino txikiagoa baldin bada ere, errore-mezua aurkeztu beharko da.

Adibideak:

desk_lag 8 2= [2, 2, 2]	desk_lag 8 4= []	desk_lag 7 3 = [7]
desk_lag 12 3= [3]	desk_lag 15 4= [5]	desk_lag 7 10 = []
desk_lag 100 3 = [5, 5]	desk_lag 75 4 = [5, 5]	

3) leh_zerrenda

Lehenengo n zenbaki lehenez osatutako zerrenda kalkulatzen duen *leh_zerrenda* funtzioa definitu murgilketaren teknika erabiliz. n-ren balioa ≥ 0 ez bada, errore mezua aurkeztu beharko da. Zenbaki bat lehena al den erabakitzeko, aurretik definituta dagoen *lehena* izeneko funtzioa erabili behar da.

Adibideak:
leh_zerrenda 5 = [2, 3, 5, 7, 11]
leh_zerrenda 0 = []

Helburua *leh_zerrenda* funtzioa definitzeko *leh_zerrenda_lag* beste funtzio bat definitzea da. Hor *leh_zerrenda_lag* funtzioak, n eta bm bi zenbaki oso emanda, bm-tik abiatuta lehenengo n zenbaki lehenez osatutako zerrenda kalkulatu beharko du. Emandako n balioa 0 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da eta bm balioa 2 baino txikiagoa baldin bada ere, errore-mezua aurkeztu beharko da.

Adibideak:
leh_zerrenda_lag 5 2= [2, 3, 5, 7, 11]
leh_zerrenda_lag 5 6= [7, 11, 13, 17, 19]
leh_zerrenda_lag 5 7= [7, 11, 13, 17, 19]
leh_zerrenda_lag 0 6= []

4) tx_zerrenda

n zenbakia baino txikiagoak diren zenbaki lehenez osatutako zerrenda kalkulatzen duen *tx_zerrenda* funtzioa definitu murgilketaren teknika erabiliz. n-ren balioa ≥ 2 ez bada, errore mezua aurkeztu beharko da. Zenbaki bat lehena al den erabakitzeko, aurretik definituta dagoen *lehena* izeneko funtzioa erabili behar da.

Adibideak:
tx_zerrenda 5 = [2, 3]
tx_zerrenda 2 = []

Helburua *tx_zerrenda* funtzioa definitzeko *tx_zerrenda_lag* beste funtzio bat definitzea da. Hor *tx_zerrenda_lag* funtzioak, n eta bm bi zenbaki oso emanda, bm-ren berdinak edo handiagoak eta n baino txikiagoak diren zenbaki lehenez osatutako zerrenda kalkulatu beharko du. Emandako n balioa 2 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da eta bm balioa 2 baino txikiagoa baldin bada ere, errore-mezua aurkeztu beharko da.

Adibideak:
tx_zerrenda_lag 5 2= [2, 3]
tx_zerrenda_lag 5 6= []
tx_zerrenda_lag 2 2 = []

5) biber (2012-13 – 0,500 puntu)

Datu bezala x zenbaki oso bat emanda, x balioa 2 zenbakiaren berredura al den erabakitzen duen *biber* funtzioa definitu murgilketa erabiliz. x zenbakiaren balioa 0 edo txikiagoa baldin bada, errore-mezua aurkeztu beharko 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 oso bat existitzen bada. Esate baterako, 8 zenbakia 2ren berredura da $8 = 2^3$ betetzen delako, baina 10 ez da 2ren berredura $10 \neq 2^k$ betetzen baita ≥ 0 den edozein k balio osorentzat. Aipatzekoa da 1 zenbakia ere 2ren berredura dela, $1 = 2^0$ betetzen baita.

```
biber :: Integer −> Bool
biber x …
```

Adibideak:
biber 8 = True
biber 10 = False

Helburua *biber* funtzioa definitzeko *biber_lag* beste funtzio bat definitzea da. Hor *biber_lag* funtzioak, x eta p bi zenbaki oso emanda, $x = p * (2^k)$ betearazten duen eta 0 edo handiagoa den k zenbaki osorik existitzen 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 :: Integer −> Integer −> Bool
biber_lag x p …
```

Adibideak:

biber_lag 32 4 = True	$32 = 4 * (2^3)$ delako
biber_lag 32 9 = False	0 edo handiagoa den eta $32 = 9 * (2^k)$ betetzen duen k zenbaki osorik ez baita existitzen
biber_lag 40 1 = False	0 edo handiagoa den eta $40 = 1 * (2^k)$ betetzen duen k zenbaki osorik ez baita existitzen
biber_lag 40 5 = True	$40 = 5 * (2^3)$ baita

biber_lag funtzioa *biber* funtzioa baino orokorragoa 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 bukatu behar da. Baina p-ren balioa x baino handiagoa izatera iristen bada, x eta p berdinak izateko aukerarik ez dagoenez False itzuli beharko da. Esate baterako, *biber_lag 32 4* deiak honako dei-sekuentzia eragingo luke:

biber_lag 32 4 * 2, biber_lag 32 4 * 2 * 2, biber_lag 32 4 * 2 * 2 * 2

Kasu honetan justu 32 lortuko litzateke eta True itzuliko litzateke. *biber_lag 32 9* deiaren kasuan, honako sekuentzia edukiko genuke:

biber_lag 32 9 * 2, biber_lag 32 9 * 2 * 2

Kasu honetan 9 * 2 * 2 balioa 32tik pasatu egin denez, False itzuliko litzateke.

6) gehiagotan (2012-13 – 0,500 puntu)

Datu bezala x eta v bi zenbaki oso eta zenbaki osozko s zerrenda bat emanda, x zenbakiaren s zerrendako agerpen-kopurua v baino handiagoa al den erabakitzen duen *gehiagotan* funtzioa definitu.

```
gehiagotan :: Int −> Int −> [Int] −> Bool
gehiagotan x v s . . .
```

Adibideak:

gehiagotan 8 3 [8, 5, 12, 8, 1] = False
8ren agerpen-kopurua ez delako 3 baino handiagoa

gehiagotan 8 3 [8, 5, 12, 8, 8, 1, 8] = True
8ren agerpen-kopurua 3 baino handiagoa delako

Helburua *gehiagotan* funtzioa definitzeko *gehiagotan_lag* funtzioa definitzea da. Hor *gehiagotan_lag* funtzioak, x eta v bi zenbaki oso, s zenbaki osozko zerrenda eta zenb zenbaki oso bat emanda, x zenbakiaren s zerrendako agerpen-kopurua gehi zenb balioa v baino handiagoa bada True eta bestela False itzuliko du.

```
gehiagotan_lag :: Int −> Int −> [Int] −> Int −> Bool
gehiagotan_lag x v s zenb …
```

Adibideak:

gehiagotan_lag 8 10 [8, 5, 12, 8, 1] 3 = False
8ren agerpen-kopurua (hau da, 2) gehi 3 ez delako 10 baino handiagoa

gehiagotan_lag 8 10 [8, 5, 12, 8, 1] 9 = True
10 balioa 8ren agerpen-kopurua (hau da, 2) gehi 9 baino txikiagoa delako

gehiagotan_lag funtzioa *gehiagotan* funtzioa baino orokorragoa da. *gehiagotan_lag x v* s *zenb* kalkulatzeko urratsak honako hauek dira:

- zenb balioa v baino handiagoa bada, True itzuli emaitza bezala.
- Bestela, s hutsa baldin bada, (eta zenb ez da v baino handiagoa), False itzuli emaitza bezala.
- Bestela, s-ko lehenengo elementua x balioaren berdina bada, hurrengo dei errekurtsiboan x eta v berdin mantenduko dira, s-ren lekuan s-ko ezkerreko ertzeko elementua kenduz lortzen den zerrenda ipini beharko da eta zenb-en ordez zenb + 1 ipini beharko da.

- Bestela, s-ko lehenengo elementua x balioaren berdina ez bada, s-ren lekuan s-ko ezkerreko ertzeko elementua kenduz lortzen den zerrenda ipini beharko da eta zenb berdin mantenduko da. Adibidez, *gehiagotan_lag 8 10 [8, 5, 12, 8, 1] 9* deiak honako dei-sekuentzia eragingo luke:

gehiagotan_lag 8 10 [8, 5, 12, 8, 1] 9 = gehiagotan_lag 8 10 [5, 12, 8, 1] 10 =
= gehiagotan_lag 8 10 [12, 8, 1] 10 = gehiagotan_lag 8 10 [8, 1] 10 =
= gehiagotan_lag 8 10 [1] 11 = True

gehiagotan x v s definitzeko *gehiagotan_lag* erabiltzerakoan, *zenb* parametroa 0 balioaz ordezkatu beharko da zenbatzaile bezala erabiliko baita.

7) hand_mr (2012-13 – 0,500 puntu)

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

```
hand_mr :: [Int] −> Int
hand_mr z …
```

Adibideak:

```
hand_mr [8, 5, 12, 7, 12] = 12

hand_mr [9] = 9
```

Helburua *hand_mr* funtzioa definitzeko *hand_mr_lag* funtzioa definitzea da. Hor *hand_mr_lag* funtzioak, 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 :: [Int] −> Int −> Int
hand_mr_lag s h . . .
```

Adibideak:

hand_mr_lag [8, 5, 12, 7, 12, 3] 6 = 12
8, 5, 12, 7, 12, 3 eta 6 balioak hartuz, handiena 12 delako

hand_mr_lag [8, 5, 12, 7, 12, 3] 20 = 20
8, 5, 12, 7, 12, 3 eta 20 balioak hartuz, handiena 20 delako

hand_mr_lag [8, 5, 12, 7, 12, 3] 5 = 12
8, 5, 12, 7, 12, 3 eta 5 balioak hartuz, handiena 12 delako

hand_mr_lag [] 17 = 17
bakarrik 17 daukagunez, handiena 17 da

hand_mr_lag funtzioa *hand_mr* funtzioa baino orokorragoa da. *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. Adibidez, *hand_mr_lag [8, 5, 12, 7, 12, 3] 6* deiak honako dei-sekuentzia eragingo luke:

  hand_mr_lag [8, 5, 12, 7, 12, 3] 6 = hand_mr_lag [5, 12, 7, 12, 3] 8 =
  = hand_mr_lag [12, 7, 12, 3] 8 = hand_mr_lag [7, 12, 3] 12 =
  = hand_mr_lag [12, 3] 12 = hand_mr_lag [3] 12 = hand_mr_lag [] 12
  = 12

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

8) bg (bikoiti gehiago) (2012-13 – 0,500 puntu)

Murgilketaren teknika erabiliz, zenbaki osoz osatutako s zerrenda bat emanda, s zerrendan bikoiti-kopurua bakoiti-kopurua baino handiagoa baldin bada True eta bestela False itzultzen duen *bg* funtzioa definitu. s zerrenda hutsa baldin bada, False itzuli beharko da.

```
bg :: [Int] −> Bool
bg s . . .
```

Adibideak:

```
bg [8, 5, 12, 7, 12] = True
bg [9, 7, 10] = False
```

Murgilketaren teknika erabiltzeko *bg_lag* funtzioa definitu behar da. Funtzio honek zenbaki osozko s zerrenda bat eta *bik* eta *bak* bi zenbaki oso emanda, s zerrendako bikoiti-kopurua gehi bik balioa s zerrendako bakoiti-kopurua gehi bak balioa baino handiagoa baldin bada True eta bestela False itzuliko du.

```
bg_lag :: [Int] −> Int −> Int −> Bool
bg_lag s bik bak …
```

Adibideak:

```
bg_lag [8, 7, 12, 10, 20, 9] 5 3 = True
```

bikoiti-kopurua (4) gehi 5, bakoiti-kopurua (2) gehi 3 baino handiagoa delako.

bg_lag [8, 7, 12, 10, 20, 9] 5 16 = False
bikoiti-kopurua (4) gehi 5, bakoiti-kopurua (2) gehi 16 baino txikiagoa delako.

bg_lag [] 5 3 = True
bikoiti-kopurua (0) gehi 5, bakoiti-kopurua (0) gehi 3 baino handiagoa delako.

bg_lag funtzioa *bg* funtzioa baino orokorragoa da.

Adibidez, *bg_lag [8, 7, 12, 10, 20, 9] 5 3* kasuan honako urratsak emango lirateke errekurtsiboki:

bg_lag [8, 7, 12, 10, 20, 9] 5 3 = bg_lag [7, 12, 10, 20, 9] 6 3 =
= bg_lag [12, 10, 20, 9] 6 4 = bg_lag [10, 20, 9] 7 4 =
= bg_lag [20, 9] 8 4 = bg_lag [9] 9 4 = bg_lag [] 10 5 = True

bg_lag erabiliz *bg* s definitzerakoan, bik eta bak parametroak zenbaki bikoitiak eta bakoitiak zenbatzeko erabili behar dira hurrenez hurren.

D) Bukaerako errekurtsibitatea murgilketaren bidez

1) fakt_be (faktoriala parametro berri batekin)

0 baino handiagoa edo berdina den x zenbakiaren faktoriala kalkulatzen duen fakt_be funtzioa definitu nahi da. Beraz, x parametroaren balioa 1 edo handiagoa baldin bada, [1..x] tarteko elementuen biderkadura itzuli behar da eta, kasu berezi bezala, x parametroaren balioa 0 baldin bada, emaitza 1 izango da. Emandako x balioa 0 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
fakt_be 4 = 24        fakt_be 3 = 6
fakt_be 0 = 1         fakt_be 5 = 120
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta bi parametro (x eta bid) izango dituen fakt_be_lag funtzioa definituko da laguntzaile bezala. Bigarren parametroa [1..x] tarteko zenbakien biderkadura gordetzeko erabiliko da. Definitu beharreko fakt_be_lag funtzioak [1..x] tarteko elementuen biderkadura bider bid itzuliko du emaitza bezala. Biderkadura hori kalkulatzeko [1..x] tarteko zenbakiak atzeraka zeharkatu beharko dira, hau da, x-tik hasi eta 1eraino. Kasu berezi bezala, x parametroaren balioa 0 baldin bada, emaitza bid izango da eta 0 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
fakt_be_lag 4 1 = 24        fakt_be_lag 4 2 = 48
fakt_be_lag 0 5= 5          fakt_be_lag 5 3= 360
```

2) fakt2_be (faktoriala bi parametro berrirekin)

0 baino handiagoa edo berdina den x zenbakiaren faktoriala kalkulatzen duen fakt2_be funtzioa definitu nahi da. Beraz, x parametroaren balioa 1 edo handiagoa baldin bada, [1..x] tarteko elementuen biderkadura itzuli behar da eta, kasu berezi bezala, x parametroaren balioa 0 baldin bada, emaitza 1 izango da. Emandako x balioa 0 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
fakt2_be 4 = 24        fakt2_be 3 = 6
fakt2_be 0 = 1         fakt2_be 5 = 120
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta hiru parametro (x, bm eta bid) izango dituen fakt2_be_lag funtzioa definituko da laguntzaile bezala. Bigarren parametroa [1..x] tarteko zenbakiak zeharkatzeko erabiliko da eta hirugarren parametroa [1..x] tarteko zenbakien biderkadura gordetzeko erabiliko da. Definitu beharreko fakt2_be_lag funtzioak [bm..x] tarteko elementuen biderkadura bider bid itzuliko du emaitza bezala. Biderkadura hori kalkulatzeko [bm..x] tarteko zenbakiak aurreraka zeharkatu beharko dira, hau da, bm-tik hasi eta x balioraino. Kasu berezi bezala, x parametroaren balioa 0 baldin bada edo bm balioa x baino handiagoa bada,

emaitza bid izango da. Beste aldetik, x parametroaren balioa 0 baino txikiagoa baldin bada edo bm parametroaren balioa 1 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
fakt2_be_lag 4 1 1 = 24      [1..4] tarteko denak bider 1
fakt2_be_lag 4 1 2 = 48      [1..4] tarteko denak bider 2
fakt2_be_lag 4 1 6 = 144     [1..4] tarteko denak bider 6

fakt2_be_lag 4 2 1 = 24      [2..4] tarteko denak bider 1
fakt2_be_lag 4 3 1 = 12      [3..4] tarteko denak bider 1
fakt2_be_lag 4 3 2 = 24      [3..4] tarteko denak bider 2
fakt2_be_lag 4 3 6 = 72      [3..4] tarteko denak bider 6
fakt2_be_lag 4 10 7 = 7      [10..4] tarte hutsa da
```

3) luzera_be (luzera parametro berri batekin)

t motako elementuz osatutako zerrenda bat emanda, zerrendako elementu-kopurua, hau da, zerrendaren luzera kalkulatzen duen luzera_be funtzioa definitu nahi da. Zerrenda hutsa baldin bada, 0 itzuliko da emaitza bezala.

Adibideak:

```
luzera_be [5, 9, 8, 1] = 4        luzera_be [] = 0
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta bi parametro izango dituen luzera_be_lag funtzioa definituko da laguntzaile bezala. Lehenengo parametroa t motako zerrenda bat izango da eta bigarren parametroa zenbaki oso bat izango da. Definitu beharreko luzera_be_lag funtzioak zerrendaren luzera gehi bigarren parametroaren balioa itzuliko du. Bigarren parametro berri hori berez zenbatzaile bezala erabiltzeko da.

Adibideak:

```
luzera_be_lag [5, 9, 8, 1] 0 = 4      luzera_be_lag [5, 9, 8, 1] 8 = 12
luzera_be_lag [] 0 = 0                luzera_be_lag [] 6 = 6
```

4) aldiz_be (elementua zerrendan zenbat aldiz agertzen den)

t motako elementu bat eta t motako zerrenda bat emanda, elementu hori zerrendan zenbat aldiz agertzen den zenbatzen duen funtzioa definitu nahi da.

Adibideak:

```
aldiz_be 8 [3, 7, 8, 20, 8, 7] = 2
aldiz_be 10 [3, 7, 8, 20, 8, 7] = 0
aldiz_be 6 [] = 0
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta hiru parametro izango dituen aldiz_be_lag funtzioa definituko da laguntzaile bezala. Lehenengo bi parametroak t motako elementu bat eta t motako zerrenda bat izango dira eta hirugarren parametroa Int motako elementu bat izango da. Definitu beharreko aldiz_be_lag funtzioak lehenengo parametroak adierazten duen balioa zerrendan zenbat aldiz agertzen den gehi hirugarren parametroaren

balioa itzuliko du. Hirugarren parametro berri hori berez agerpen-kopurua zenbatzeko da.

Adibideak:
aldiz_be_lag 8 [3, 7, 8, 20, 8, 7] 0 = 2
aldiz_be_lag 8 [3, 7, 8, 20, 8, 7] 6 = 8
aldiz_be_lag 10 [3, 7, 8, 20, 8, 7] 0 = 0
aldiz_be_lag 10 [3, 7, 8, 20, 8, 7] 4 = 4
aldiz_be_lag 6 [] 0 = 0
aldiz_be_lag 6 [] 3 = 3

5) gainditu_be (zenbakia gainditzeko adineko batura duen aurrizki laburrenaren luzera)

Osoa den n zenbaki bat eta zenbaki osozko zerrenda bat emanda, zerrenda horri dagokionez, n balioa gainditzeko adineko batura duen aurrizki laburrenaren luzera kalkulatuko duen gainditu_be funtzioa definitu nahi da. Aurrizki hutsaren, hau da, zerrenda hutsaren batura 0 dela kontuan hartu beharko da. Zerrendako elementuak batuz n ezin bada gainditu, -1 itzuliko da emaitza gisa.

Adibideak:

gainditu_be 17 [5, 9, 8, 1] = 3	gainditu_be (-4) [5, 9, 8, 1] = 0
gainditu_be 2 [5, 9, 8, 1] = 1	gainditu_be 80 [5, 9, 8, 1] = -1
gainditu_be 5 [5, 9, 8, 1] = 2	gainditu_be 14 [5, 9, 8, 1] = 3
gainditu_be 7 [5, -2, 8, 1] = 3	gainditu_be (-4) [-8, 1, -8, 5] = 0

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta lau parametro izango dituen gainditu_be_lag funtzioa definituko da laguntzaile gisa. Lehenengo bi parametroak osoa den n zenbaki bat eta zenbaki osozko zerrenda bat izango dira. Hirugarren eta laugarren parametroak beste bi zenbaki oso izango dira. Definitu beharreko gainditu_be_lag funtzioak, bigarren parametrotzat emandako zerrendari dagokionez, n gainditzeko hirugarren parametroari batu beharreko aurrizki laburrenaren luzera kalkulatuko du eta luzera hori gehi laugarren parametroaren balioa itzuliko du. Aurrizki hutsaren, hau da, zerrenda hutsaren batura 0 dela kontuan hartu beharko da. Zerrendako elementuak eta hirugarren parametroa batuz n ezin bada gainditu, -1 itzuliko da emaitza gisa. Hirugarren parametro berri hori berez batura kalkulatuz joateko da eta laugarren parametroa zenbatzaile gisa erabiltzeko da.

Adibideak:

gainditu_be_lag 17 [5, 9, 8, 1] 0 0 = 3	gainditu_be_lag 5 [5, 9, 8, 1] 0 0 = 2
gainditu_be_lag 17 [5, 9, 8, 1] 0 4 = 7	gainditu_be_lag 5 [5, 9, 8, 1] 0 7 = 9
gainditu_be_lag 17 [5, 9, 8, 1] 10 0 = 2	gainditu_be_lag 7 [5, -2, 8, 1] 0 0 = 3
gainditu_be_lag 2 [5, 9, 8, 1] 24 0 = 0	gainditu_be_lag 7 [5, -2, 8, 1] 6 0 = 1
gainditu_be_lag 2 [5, 9, 8, 1] 24 6 = 6	gainditu_be_lag 7 [5, -2, 8, 1] 6 (-3) = -2
gainditu_be_lag 2 [5, 9, 8, 1] 24 (-6) = -6	gainditu_be_lag (-4) [5, 9, 8, 1] 2 7= 7
gainditu_be_lag 2 [5, 9, 8, 1] 0 0 = 1	gainditu_be_lag (-4) [5, 9, 8, 1] (-10) 1= 3
gainditu_be_lag 2 [5, 9, 8] (-4) (-9) = -7	gainditu_be_lag (-4) [-8, 1, -8] (-10) 3= -1

6) lehenpos_be (lehenengo agerpenaren posizioa)

Osoa den n zenbaki bat eta zenbaki osoz osatutako zerrenda bat emanda, n-ren lehenengo agerpenaren posizioa (ezkerretik hasita) itzultzen duen lehenpos_be funtzioa definitu nahi da. Zenbakia zerrendan ez bada agertzen, 0 balioa itzuli behar da.

Adibideak:

```
lehenpos_be 8 [4, 9, 8, 17, 8] = 3
lehenpos_be 8 [7, 3, 0, 0, 97] = 0
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta hiru parametro izango dituen lehenpos_be_lag funtzioa definituko da laguntzaile gisa. Lehenengo bi parametroak osoa den n zenbaki bat eta zenbaki osozko zerrenda bat izango dira eta hirugarren parametroa beste zenbaki oso bat izango da. Definitu beharreko lehenpos_be_lag funtzioak lehenengo parametroak (hau da n datuak) adierazten duen balioaren lehenengo agerpenaren posizioa (ezkerretik hasita) gehi hirugarren parametroaren balioa itzuliko du. Kasu berezi gisa, lehenengo parametroak adierazten duen n balioa zerrendan ez bada agertzen, 0 balioa itzuli beharko da. Hirugarren parametro berri hori berez zenbatzaile gisa erabiltzeko da.

Adibideak:

```
lehenpos_be_lag 8 [4, 9, 8, 17, 8] 0 = 3
lehenpos_be_lag 8 [4, 9, 8, 17, 8] 12 = 15
lehenpos_be_lag 8 [7, 3, 0, 0, 97] 0 = 0
lehenpos_be_lag 8 [7, 3, 0, 0, 97] 2 = 0
```

7) ak_be (azkena kendu bukaerako errekurtsibitatearekin)

t motako elementuz osatutako zerrenda bat emanda, azkeneko elementua (eskuineko ertzean dagoena) kenduz gelditzen den zerrenda itzultzen duen funtzioa definitu nahi da. Zerrenda hutsa bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
ak_be [4, 8, 3, 5, 2] = [4, 8, 3, 5]
ak_be [] = error "Zerrenda hutsa."
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta bi parametro izango dituen ak_be_lag funtzioa definituko da laguntzaile bezala. Parametro biak t motako zerrendak izango dira. Definitu beharreko ak_be_lag funtzioak honako bi zerrenda hauek elkartuz lortuko den zerrenda itzuliko du:

1. Datu bezala emandako bigarren zerrenda
2. Datu bezala emandako lehenengo zerrendatik azkeneko elementua (eskuineko ertzean dagoena) kenduz geratzen den zerrenda

Datu bezala emandako lehenengo zerrenda hutsa bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
ak_be_lag [4, 8, 3, 5, 2] [] = [4, 8, 3, 5]
ak_be_lag [4, 8, 3, 5, 2] [6, 7] = [6,7, 4, 8, 3, 5]
ak_be_lag [] [] = error "Lehenengo zerrenda hutsa da."
ak_be_lag [] [6, 7] = error "Lehenengo zerrenda hutsa da."
```

8) bz_be (bi zerrendatako posizio bereko elementuak batuz zerrenda)

Zenbaki osoz osatutako bi zerrenda emanda, posizio berean dauden elementuen baturez osatutako zerrenda itzultzen duen funtzioa definitu nahi da. Zerrenden luzera desberdina bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
bz_be [4, 8, 3, 5] [1, 0, 6, 2] = [5, 8, 9, 7]
bz_be [] [] = []
bz_be [] [4, 5, 2] = error "Luzera desberdina."
```

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta hiru parametro izango dituen bz_be_lag funtzioa definituko da laguntzaile bezala. Hiru parametroak zenbaki osozko zerrendak izango dira. Definitu beharreko bz_be_lag funtzioak honako bi zerrenda hauek elkartuz lortzen den zerrenda itzuli beharko du:

1. Datu bezala emandako hirugarren zerrenda eta
2. Datu bezala emandako lehenengo bi zerrendatan posizio berean dauden elementuen baturez osatutako zerrenda

Datu bezala emandako lehenengo bi zerrenden luzera desberdina bada, errore-mezua aurkeztu beharko da.

Adibideak:

```
bz_be_lag [4, 8, 3, 5] [1, 0, 6, 2] [] = [5, 8, 9, 7]
bz_be_lag [4, 8, 3, 5] [1, 0, 6, 2] [11, 12] = [11, 12, 5, 8, 9, 7]
bz_be_lag [] [] [] = []
bz_be_lag [] [] [11, 12] = [11, 12]
bz_be_lag [] [4, 5, 2] [11, 12] = error "Luzera desberdina."
```

9) th_be (txikiagoak ezkerraldera eta handiagoak eskuinaldera)

Osoa den n zenbaki bat eta zenbaki osozko zerrenda bat emanda, n baino txikiagoak edo berdinak direnak ezkerraldean eta n baino handiagoak direnak eskuinaldean ipiniz osatutako zerrenda itzultzen duen funtzioa definitu nahi da. Txikiagoak edo berdinak direnek beraien artean hasierako ordenari eutsi beharko diote eta handiagoak direnek ere beraien artean hasierako ordenari eutsi beharko diote. Gerta daiteke n balioa zerrendan ez agertzea. Datu bezala emandako zerrenda hutsa bada, zerrenda hutsa aurkeztu beharko da.

Adibideak:
th_be 4 [4, 8, 3, 5, 2, 1, 12] = [4, 3, 2, 1, 8, 5, 12]
th_be 5 [3, 2, 4] = [3, 2, 4]
th_be 5 [] = []

Bukaerako errekurtsibitatea eduki nahi denez, murgilketa erabiliko da eta lau parametro izango dituen th_be_lag funtzioa definituko da laguntzaile bezala: osoa den n zenbaki bat eta zenbaki osozko hiru zerrenda. Definitu beharreko th_be_lag funtzioak honako lau zerrenda elkartuz lortzen den zerrenda itzuli beharko du:

1. Hirugarren parametro bezala emandako zerrenda.
2. Bigarren parametro bezala emandako zerrendan n baino txikiagoak edo berdinak diren elementuez osatutako zerrenda. Elementu horiek beraien arteko hasierako ordenari eutsi beharko diote.
3. Laugarren parametro bezala emandako zerrenda.
4. Bigarren parametro bezala emandako zerrendan n baino handiagoak diren elementuez osatutako zerrenda. Elementu horiek beraien arteko hasierako ordenari eutsi beharko diote.

Bigarren parametro bezala emandako zerrenda hutsa baldin bada, hirugarren eta laugarren parametroak elkartuz lortzen den zerrenda itzuli beharko da.

Bigarren zerrenda (hirugarren parametroa) n baino txikiagoak edo berdinak diren elementuak gordetzeko erabiliko da eta hirugarren zerrenda (laugarren parametroa) n baino handiagoak diren elementuak gordetzeko erabiliko da.

Adibideak:
th_be_lag 4 [4, 8, 3, 5, 2, 1, 12] [] [] = [4, 3, 2, 1, 8, 5, 12]
th_be_lag 4 [4, 8, 3, 5, 2, 1, 12] [6, 7] [11, 0] =
[6, 7, 4, 3, 2, 1, 11, 0, 8, 5, 12]
th_be_lag 4 [] [] [] = []
th_be_lag 4 [][6, 7] [11, 0] = [6, 7, 11, 0]

E) Zerrenda eraketa

1) lehenengo_lehenak (lehenengo n zenbaki lehen)

Osoa den n zenbaki oso bat emanda, lehenengo n zenbaki lehenez osatutako zerrenda itzultzen duen lehenengo_lehenak izeneko funtzioa definitu nahi da. Sarrera bezala emandako zenbakia negatiboa baldin bada, errore-mezua aurkeztu beharko da.

Adibideak:
lehenengo_lehenak 5 = [2, 3, 5, 7, 11]
lehenengo_lehenak 0 = []
lehenengo_lehenak (-3) = error "Negatiboa"

Aukera bat, **Integer** mota, aurredefinitutako **genericTake** funtzioa eta **Zerrenda_eraketa.hs** moduluan definituta dagoen **lehena_ze** funtzioa erabiltzea da.

2) lehen_denak

Zenbaki lehen denez osatutako zerrenda infinitua aurkeztuz joango den lehen_denak izeneko funtzioa definitu nahi da.

Adibidea:
lehen_denak = [2, 3, 5, 7, 11, 13, 17, 19, 23, ...]

Aukera bat, **Integer** mota eta **Zerrenda_eraketa.hs** moduluan definituta dagoen **lehena_ze** funtzioa erabiltzea da.

3) lehen_txikiagoak

Osoa den n zenbaki oso bat emanda, n baino txikiagoak diren zenbaki lehenez osatutako zerrenda itzultzen duen lehen_txikiagoak izeneko funtzioa definitu nahi da. Sarrera bezala emandako zenbakia 0 edo negatiboa baldin bada, zerrenda hutsa aurkeztu beharko da.

Adibideak:
lehen_txikiagoak 20 = [2, 3, 5, 7, 11, 13, 17, 19]
lehen_txikiagoak 19 = [2, 3, 5, 7, 11, 13, 17]
lehen_txikiagoak (-2) = []

Aukera bat, **Integer** mota eta **Zerrenda_eraketa.hs** moduluan definituta dagoen **lehena_ze** funtzioa erabiltzea da.

4) lehen_zenbatuak

Zenbaki lehen bakoitzaz eta zenbaki hori zenbaki lehenen zerrendan zenbatgarrena den adierazten duen zenbakiaz osatutako bikoteen zerrenda infinitua aurkeztuz joango den lehen_zenbatuak izeneko funtzioa definitu nahi da.

Adibidea:
lehen_zenbatuak = [(1, 2), (2, 3), (3, 5), (4, 7), (5, 11), (6, 13), (7, 17), (8, 19), (9, 23), (10, 29), ...]

Aukera bat, **Integer** mota, aurredefinitutako **zip** funtzioa eta **2. ariketa**n definitutako **lehen_denak** funtzioa erabiltzea da.

5) lehenengo_lehen_zenbatuak

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 lehenengo_lehen_zenbatuak izeneko funtzioa definitu nahi da. Emandako n balioa negatiboa baldin bada, errore-mezua aurkeztu beharko da.

Adibidea:
lehenengo_lehen_zenbatuak 5 = [(1, 2), (2, 3), (3, 5), (4, 7), (5, 11)]

Aukera bat, **Integer** mota, aurredefinitutako **genericTake** funtzioa eta **4. ariketa**n definitutako **lehen_zenbatuak** funtzioa erabiltzea da.

6) lehenak_nondik

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 lehenak_nondik izeneko funtzioa definitu nahi da. Emandako n balioa negatiboa bada edo p positiboa ez bada, errore-mezua aurkeztu beharko da.

Adibidea:
lehenak_nondik 5 7 = [(7, 17), (8, 19), (9, 23), (10, 29), (11, 31)]

Aukera bat, **Integer** mota, aurredefinitutako **genericTake** funtzioa eta **4. ariketa**n definitutako **lehen_zenbatuak** funtzioa erabiltzea da.

Beste aukera bat, **Integer** mota, aurredefinitutako **genericTake** eta **genericDrop** funtzioak eta **4. ariketa**n definitutako **lehen_zenbatuak** funtzioa erabiltzea da.

7) lehena_aukeratu

Osoa den p zenbaki bat emanda, p-garren zenbaki lehena itzultzen duen lehena_aukeratu izeneko funtzioa definitu nahi da. Emandako p balioa 0 edo negatiboa baldin bada, errore-mezua aurkeztu beharko da.

Adibidea:
lehena_aukeratu 10 = 29

Aukera bat, **Integer** mota, aurredefinitutako **genericDrop** funtzioa eta **1. ariketa**n definitutako **lehenengo_lehenak** funtzioa erabiltzea da.

8) lehenak_aukeratu1

Zenbaki osozko s zerrenda bat emanda, s zerrendan dauden zenbakiez eta zenbaki horiei dagozkien zenbaki lehenez osatutako bikoteak dituen zerrenda itzultzen duen lehenak_aukeratu1 izeneko funtzioa definitu nahi da. Bikoteak posizioaren arabera ordenatuta egon behar dute. Gainera, s zerrendan balio bat errepikatuta agertzen bada ere, balio horri dagokion bikoteak behin bakarrik agertu beharko du. Sarrerako zerrenda hutsa bada edo balio positiborik ez badu, zerrenda hutsa itzuli beharko da. 0 eta zenbaki negatiboak ez dira kontuan hartuko.

Adibidea:
lehenak_aukeratu1 [3, 6, -5, 10, 8, 6] = [(3, 5), (6, 13), (8, 19), (10, 29)]

Zerrendan 10 balioa 8 baino lehenago agertu arren, bikoteen zerrendan 8ri dagokion bikotea lehenago agertzen da.

Aukera bat, **Integer** mota, aurredefinitutako **`elem`**, **genericTake** eta **maximum** funtzioak eta **4. ariketa**n definitutako **lehen_zenbatuak** funtzioa erabiltzea da.

9) lehenak_aukeratu2

Zenbaki osozko s zerrenda bat emanda, s zerrendan dauden zenbakiez eta zenbaki horiei dagozkien zenbaki lehenez osatutako bikoteak dituen zerrenda itzultzen duen lehenak_aukeratu2 izeneko funtzioa definitu nahi da. Bikoteak s zerrendako ordena jarraituz egon behar dute. Gainera, s zerrendan balio bat errepikatuta agertzen bada ere, balio horri dagokion bikoteak ere errepikatuta agertu beharko du. Sarrerako zerrenda hutsa bada edo balio positiborik ez badu, zerrenda hutsa itzuli beharko da. 0 eta zenbaki negatiboak ez dira kontuan hartuko.

Adibidea:
lehenak_aukeratu2 [3, 6, -5, 10, 8, 6] = [(3, 5), (6, 13), (10, 29), (8, 19), (6, 13)]

Hasierako zerrendan 10 balioa 8 baino lehenago agertzen denez, bikoteen zerrendan ere 10 balioari dagokion bikotea lehenago agertzen da.

Aukera bat, **Integer** mota, aurredefinitutako **zip** funtzioa eta **7. ariketa**n definitutako **lehena_aukeratu** funtzioa erabiltzea da.

10) lehenaren_posizioa

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

Adibideak:
lehenaren_posizioa 11 = 5
lehenaren_posizioa 7 = 4
lehenaren_posizioa 8 = error "Ez da lehena"

Aukera bat, **Integer** mota, **Zerrenda_eraketa.hs** moduluan definituta dagoen **lehena_ze** funtzioa eta aurredefinitutako **genericLength** eta **takeWhile** funtzioak erabiltzea da.

11) batura_bera

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

Adibideak:
batura_bera 0 = [(0, 0)]
batura_bera 1 = [(0, 1), (1, 0)]
batura_bera 2 = [(0, 2), (1, 1), (2, 0)]
batura_bera 3 = [(0, 3), (1, 2), (2, 1), (3, 0)]

12) n_bider_n

N × N multzoko bikoteak ordenatzeko aukera egokiena honako hau zela ikusi genuen 2. gaian:

[(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), …]

Zerrenda horretan, hasteko, batura bezala 0 ematen duten bikoteak ditugu, gero batura bezala 1 ematen dutenak, gero 2 ematen dutenak eta abar.

(0,0) → 0
(0,1) (1,0) → 1
(0,2) (1,1) (2,0) → 2
(0,3) (1,2) (2,1) (3,0) → 3
(0,4) (1,3) (2,2) (3,1) (4,0) → 4
…

Haskell-ez

```
[(x, y) | x <- [0..],y <- [0..]]
```

idazten badugu, ez dugu lortuko bikoteak guk nahi dugun ordenean agertzea. Beste orden honetan agertuko dira:

[(0,0), (0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (0,7), (0,8), (0,9), (0,10), (0,11), …

Beraz, lehenengo osagai bezala 0 elementua duten bikoteak aurkeztuz joango ginateke amaierarik gabe. Inoiz ez lirateke aurkeztuko lehenengo osagai bezala 1 zenbakia edo 0 ez den beste edozein zenbaki duen bikoterik.

Bikoteak goian aipatutako orden egokian dituen zerrenda infinitua aurkeztuz joango den ***n_bider_n*** izeneko funtzioa definitu behar da:

[(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), …]

Horretarako, aurreko ariketan definitutako **batura_bera** izeneko funtzioa eta zerrendaz osatutako zerrenda bat emanda, barneko zerrenda horiek denak elkartuz osatutako zerrenda itzultzen duen aurredefinitutako **concat** funtzioa erabiltzea da aukera bat.

13) bikote_kop

Osoa eta ≥ 0 den n zenbaki bat emanda, *n_bider_n* izeneko funtzioak sortzen duen zerrenda infinituko lehenengo n bikoteak dituen zerrenda itzultzen duen ***bikote_kop*** izeneko funtzioa definitu. Emandako n balioa negatiboa denean errore mezua aurkeztu beharko da.

Adibideak:
bikote_kop 0 = []
bikote_kop 3 = [(0,0), (0,1), (1,0)]
bikote_kop 5 = [(0,0), (0,1), (1,0), (0,2), (1,1)]

Horretarako, **aurreko ariketa**n definitutako **n_bider_n** izeneko funtzioa eta aurredefinitutako **genericTake** funtzioa erabiltzea da aukera bat.

14) bikote_zenbatuak

n_bider_n izeneko funtzioak itzultzen duen zerrendako bikote bakoitza zerrendan zenbatgarrena den adieraziz osatutako honako erako zerrenda itzultzen duen ***bikote_zenbatuak*** izeneko funtzioa definitu:

[(1, (0,0)) (2,(0,1)) (3,(1,0)) (4,(0,2)) (5,(1,1)) (6,(2,0)) (7,(0,3)) (8,(1,2)) (9,(2,1)) (10,(3,0)) (11,(0,4)) (12,(1,3)) (13,(2,2)) (14,(3,1)) (15,(4,0)), …

Horretarako, **12. ariketa**n definitutako **n_bider_n** izeneko funtzioa eta aurredefinitutako **zip** funtzioa erabiltzea da aukera bat.

15) zenbat_zenbatu

Osoa eta ≥ 0 den n zenbaki bat emanda, *bikote_zenbatuak* izeneko funtzioak itzultzen duen zerrendako lehenengo n elementuez osatutako zerrenda itzultzen duen ***zenbat_zenbatu*** izeneko funtzioa definitu. Emandako n balioa negatiboa denean errore-mezua aurkeztu beharko da:

Adibideak:
zenbat_zenbatu 0 = []
zenbat_zenbatu 3 = [(1, (0,0)) (2,(0,1)) (3,(1,0))]
zenbat_zenbatu 5 = [(1, (0,0)) (2,(0,1)) (3,(1,0)) (4,(0,2)) (5,(1,1))]

Horretarako, **aurreko ariketa**n definitutako **bikote_zenbatuak** izeneko funtzioa eta aurredefinitutako **genericTake** funtzioa erabiltzea da aukera bat.

16) ezk (2012-13)

Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, bikoteetako lehenengo osagaiez eratutako zerrenda itzuliko duen *ezk* izeneko funtzioa definitu Haskell-ez.

```
ezk :: [(Integer, Integer)] -> [Integer]
ezk s = …
```

Adibideak:
ezk [(5, 2), (3, 0), (8, 20)] = [5, 3, 8]
ezk [] = []

17) bikoiztu (2012-13)

Osoa den n zenbaki bat emanda, (0, 0)-tik (n, n)-rainoko elementu berdinez osatutako bikoteez eratutako zerrenda itzuliko duen *bikoiztu* funtzioa definitu Haskell-ez. Emandako n balioa negatiboa baldin bada, zerrenda hutsa itzuli beharko da.

```
bikoiztu :: Integer -> [(Integer, Integer)]
bikoiztu n = …
```

Adibideak:
bikoiztu 5 = [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]
bikoiztu -3 = []

18) konst (2012-13)

Osoak diren x eta n bi zenbaki emanda, x zenbakiaren n errepikapen dituen zerrenda itzultzen duen *konst* funtzioa definitu Haskell-ez. Emandako n balioa 0 baldin bada, zerrenda hutsa itzuli beharko da. Bestalde, n negatiboa baldin bada, errore-mezua aurkeztu beharko da.

```
konst :: Integer -> Integer -> [Integer]
konst x n = …
```

Adibideak:

```
konst 10 4 = [10, 10, 10, 10]
konst 10 0 = []
```

19) zero (2012-13)

Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, gutxienez osagaietako bat 0 duten bikoteez eratutako zerrenda itzuliko duen *zero* izeneko funtzioa definitu Haskell-ez.

```
zero :: [(Integer, Integer)] -> [(Integer, Integer)]
zero s = …
```

Adibideak:

```
zero [(5, 2), (3, 0), (7, 0), (8, 20), (0, 0), (0, 15), (4, 4)] =
= [(3, 0), (7, 0), (0, 0), (0, 15)]

zero [(5, 2), (4, 4), (12, 80)] = []

zero [] = []
```

20) trukatu (2012-13)

Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, osagaien ordena trukatuz lortzen diren bikoteez eratutako zerrenda itzuliko duen *trukatu* izeneko funtzioa definitu Haskell-ez.

```
trukatu :: [(Integer, Integer)] -> [(Integer, Integer)]
trukatu s = …
```

Adibideak:

```
trukatu [(5, 2), (3, 0), (8, 20), (4, 4)] = [(2, 5), (0, 3), (20, 8), (4, 4)]

trukatu [(5, 2)] = [(2, 5)]

trukatu [] = []
```

21) f (2012-13)

Zenbaki osozko s zerrenda bat emanda, s-ko elementu bikoitiak errepikatuz osatutako bikoteez eratutako zerrenda itzuliko duen *f* izeneko funtzioa definitu Haskell-ez.

```
f :: [Integer] -> [(Integer, Integer)]
f s = …
```

Adibideak:

f [4, 6, 6, 3, 20, 11] = [(4, 4), (6, 6), (6, 6), (20, 20)]
4, 6 eta 20 bikoitiak direnez, hiru balio horiek errepikatuz hiru bikote osatu dira. 3 eta 11 bakoitiak dira.

f [] = []

22) inkr_ze (2012-13)

Zenbaki osozko s zerrenda bat emanda, s-ko elementu bakoitzari 1 gehituz lortzen den zerrenda itzuliko duen *inkr_ze* izeneko funtzioa definitu Haskell-ez.

```
inkr_ze:: [Integer] -> [Integer]
inkr_ze s = …
```

Adibideak:

inkr_ze [5, 10, 3, 3, 8] = [6, 11, 4, 4, 9]

inkr_ze [] = []

23) bid_ze (2012-13)

Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, bikote bakoitzeko osagaiak bidertuz lortzen diren balioez eratutako zerrenda itzuliko duen *bid_ze* izeneko funtzioa definitu Haskell-ez.

```
bid_ze:: [(Integer, Integer)] -> [Integer]
bid_ze s = …
```

Adibideak:

bid_ze [(5, 2), (3, 0), (7, 0), (8, 2), (3, 5), (4, 4)] = [10, 0, 0, 16, 15, 16]

bid_ze [] = []

24) leh_os (2012-13)

Zenbaki osozko bikoteez osatutako s zerrenda bat emanda, bikote bakoitzeko lehenengo osagaia errepikatuz lortzen diren bikoteez eratutako zerrenda itzuliko duen *leh_os* izeneko funtzioa definitu Haskell-ez.

```
leh_os:: [(Integer, Integer)] -> [(Integer, Integer)]
leh_os s = …
```

Adibideak:

```
leh_os [(5, 2), (3, 0), (8, 20), (4, 4)] = [(5, 5), (3, 3), (8, 8), (4, 4)]
leh_os [] = []
```

25) handiagoak (2012-13)

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

```
handiagoak :: Integer -> [Integer] -> [Integer]
handiagoak n s = …
```

Adibideak:

```
handiagoak 5 [10, 4, 5, 8, 0, 10] = [10, 8, 10]
handiagoak 20 [10, 4, 5, 8, 0, 10] = []
handiagoak 5 [] = []
```

26) desberdinak (2012-13)

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

```
desberdinak :: [Integer] -> Integer -> [Integer]
desberdinak s x = …
```

Adibideak:

```
desberdinak [5, 10, 3, 3, 8] 10 = [5, 3, 3, 8]
desberdinak [5, 10, 3, 3, 8] 4 = [5, 10, 3, 3, 8]
desberdinak [5, 10, 3, 3, 8] 3 = [5, 10, 8]
desberdinak [] 4 = []
```

27) neg (2012-13)

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 *neg* funtzioa definitu Haskell-ez.

```
neg :: Integer -> [(Integer, Integer)] -> [(Integer, Integer)]
neg n s = …
```

Adibideak:

```
neg 5 [(5, 2), (3, 0), (5, -7), (8, 20), (5, -10), (0, 15), (4, 4)] =
= [(5, -7), (5, -10)]

neg 5 [(5, 2), (4, 4), (12, 80)] = []

neg 5 [] = []
```

28) inkrb (2012-13)

Zenbaki osozko bikoteez osatutako s zerrenda emanda, bikoteen osagaiei 1 balioa gehituz lortzen diren bikoteak dituen zerrenda itzultzen duen *inkrb* funtzioa definitu Haskell-ez.

```
inkrb :: [(Integer, Integer)] -> [(Integer, Integer)]
inkrb s = …
```

Adibideak:

```
inkrb [(5, 2), (3, 0), (8, 20), (4, 4)] = [(6, 3), (4, 1), (9, 21), (5, 5)]

inkrb [] = []
```

29) rep (2012-13)

Osoa den x zenbaki bat eta zenbaki osoz osatutako s zerrenda bat emanda, s zerrendan dauden x zenbakiaren errepikapenez osatutako zerrenda itzultzen duen *rep* funtzioa definitu Haskell-ez.

```
rep :: Integer -> [Integer] -> [Integer]
rep x s = …
```

Adibideak:

```
rep 4 [5, 10, 4, 4, 8] = [4, 4]
rep 4 [6, 20] = []
rep 4 [] = []
```

F) Sarrera/Irteera

1) zatitzaileak_si1

Erabiltzaileari osoa eta positiboa den zenbaki bat eskatu, zenbakia jaso eta zenbaki horren zatitzaileen zerrenda aurkeztuko duen zatitzaileak_si1 funtzioa definitu. Programaren exekuzioa bukatzen denean, programa bukatu dela esanez mezu bat ere aurkeztu beharko du. Erabiltzaileak tekleatutako zenbakia positiboa ez bada, zenbakia ez dela egokia esanez mezu bat aurkeztu eta exekuzioa bukatu egin beharko da.

Exekuzio-adibidea:

```
Positiboa den zenbaki oso bat idatzi:
14
Zatitzaileen zerrenda: [1, 2, 7, 14]
Exekuzioaren bukaera.
```

Exekuzio-adibidea:

```
Positiboa den zenbaki oso bat idatzi:
-5
Idatzitako zenbakia ez da positiboa.
Exekuzioaren bukaera.
```

Aukera bat Zerrenda_eraketa.hs moduluan definitutako zatizer_ze funtzioa erabiltzea da. Zerrenda-eraketaren teknika erabiliz ere sor daiteke zuzenean zatitzaileen zerrenda.

2) zatitzaileak_si2 eta positiboa_eskatu

Alde batetik, erabiltzaileari osoa eta positiboa den zenbaki bat eskatu, zenbakia jaso eta zenbaki hori eta zenbaki horren zatitzaileen zerrenda aurkeztuko dituen **zatitzaileak_si2** funtzioa definitu. Programaren exekuzioa bukatzen denean, programa bukatu dela esanez mezu bat ere aurkeztu beharko du.

Erabiltzaileak tekleatutako zenbakia positiboa ez bada, zenbakia ez dela egokia esanez mezu bat aurkeztu eta beste zenbaki bat eskatu beharko da, positiboa den zenbaki bat lortu arte. Eskatze-prozesu hori burutzeko, zenbaki positibo bat lortu arte zenbakia eskatzeko prozesua errepikatuko duen **positiboa_eskatu** izeneko funtzioa definitu beharko da. Funtzio horrek zenbaki positibo bat eskuratzen duenean, zenbaki hori itzuli beharko du beste funtzio batetik jaso ahal izateko moduan.

Exekuzio-adibidea:

```
Positiboa den zenbaki oso bat idatzi:
14
14 zenbakiaren zatitzaileen zerrenda: [1, 2, 7, 14]
Exekuzioaren bukaera.
```

Exekuzio-adibidea:

```
Positiboa den zenbaki oso bat idatzi:
-5
Idatzitako zenbakia ez da positiboa.
Positiboa den zenbaki oso bat idatzi:
0
Idatzitako zenbakia ez da positiboa.
Positiboa den zenbaki oso bat idatzi:
12
12 zenbakiaren zatitzaileen zerrenda: [1, 2, 3, 6, 12]
Exekuzioaren bukaera.
```

Ariketa honetan ere, aukera bat Zerrenda_eraketa.hs moduluan definitutako zatizer_ze funtzioa erabiltzea da eta beste aukera bat zerrenda-eraketaren teknika erabiliz zuzenean zatitzaileen zerrenda sortzea izango litzateke.

3) zatitzaileak_si3 eta zat_gehiago

Alde batetik, erabiltzaileari osoa eta positiboa den zenbaki bat eskatu, zenbakia jaso eta zenbaki hori eta zenbaki horren zatitzaileen zerrenda aurkeztu eta beste zenbakiren baten zatitzaileen zerrenda kalkulatu nahi al den galdetuko duen **zatitzaileak_si3** funtzioa definitu. Beste zenbakiren baten zatitzaileak kalkulatu nahi al diren galdetutakoan, erabiltzaileak b edo e (bai edo ez) erantzun beharko du. Erabiltzaileak e erantzuten badu, programaren exekuzioa bukatu dela esanez mezu bat aurkeztu beharko da. Erabiltzaileak b erantzuten badu, beste zenbaki positibo bat eskatu eta prozesu osoa behin eta berriz burutuko da, erabiltzaileak bukatzea aukeratu arte.

Erabiltzaileari beste zenbakiren baten zatitzaileak nahi al dituen galdetzeko, galdera burutu, erabiltzailearen erantzuna jaso eta erantzuna b edo e izan al den ziurtatzeaz arduratuko den **zat_gehiago** izeneko funtzioa definitu beharko da. Erabiltzaileak tekleatutakoa b edo e ez bada, erantzuna egokia ez dela esanez mezu bat aurkeztu eta berriro galdetu beharko da, b edo e lortu arte.

Era berean, erabiltzaileak tekleatutako zenbakia positiboa ez bada, zenbakia ez dela egokia esanez mezu bat aurkeztu eta beste zenbaki bat eskatu beharko da, positiboa den zenbaki bat lortu arte. Eskatze-prozesu hori burutzeko, aurreko ariketan definitutako **positiboa_eskatu** izeneko funtzioa erabili beharko da. Funtzio hori zenbaki positibo bat lortu arte zenbakia eskatzeko prozesua errepikatzeaz arduratuko da. Funtzio horrek zenbaki positibo bat eskuratzen duenean, zenbaki hori itzuli beharko du beste funtzio batetik jaso ahal izateko moduan.

Exekuzio-adibidea:

```
Positiboa den zenbaki oso bat idatzi:
14
14 zenbakiaren zatitzaileen zerrenda: [1, 2, 7, 14]
Beste zenbakiren baten zatitzaileak nahi al dituzu? (b/e):
b
Positiboa den zenbaki oso bat idatzi:
12
12 zenbakiaren zatitzaileen zerrenda: [1, 2, 3, 6, 12]
Exekuzioaren bukaera.
```

Exekuzio-adibidea:

```
Positiboa den zenbaki oso bat idatzi:
-5
Idatzitako zenbakia ez da positiboa.
Positiboa den zenbaki oso bat idatzi:
0
Idatzitako zenbakia ez da positiboa.
Positiboa den zenbaki oso bat idatzi:
12
12 zenbakiaren zatitzaileen zerrenda: [1, 2, 7, 14]
Beste zenbakiren baten zatitzaileak nahi al dituzu? (b/e):
j
Erantzuna ez da egokia...
Beste zenbakiren baten zatitzaileak nahi al dituzu? (b/e):
a
Erantzuna ez da egokia...
Beste zenbakiren baten zatitzaileak nahi al dituzu? (b/e):
s
Positiboa den zenbaki oso bat idatzi:
7
7 zenbakiaren zatitzaileen zerrenda: [1, 7]
Beste zenbakiren baten zatitzaileak nahi al dituzu? (b/e):
n
Exekuzioaren bukaera.
```

Ariketa honetan ere, aurreko bietan bezala, aukera bat Zerrenda_eraketa.hs moduluan definitutako zatizer_ze funtzioa erabiltzea da eta beste aukera bat zerrenda-eraketaren teknika erabiliz zuzenean zatitzaileen zerrenda sortzea izango litzateke. Enuntziatuan aurreko ariketan definitutako positiboa_eskatu funtzioa erabili behar dela ere esan da.

4) zatitzaileak_si4 eta zat_nahi

Alde batetik, erabiltzaileari zenbakiren baten zatitzaileen zerrenda kalkulatzea nahi al duen galdetuz hasten den **zatitzaileak_si4** funtzioa definitu. Zenbakiren baten zatitzaileen zerrenda kalkulatzea nahi al duen galdetutakoan, erabiltzaileak b edo e (bai edo ez) erantzun beharko du. Erabiltzaileak e erantzuten badu, programaren exekuzioa bukatu dela esanez mezu bat aurkeztu beharko da. Erabiltzaileak b erantzuten badu, zenbaki positibo bat eskatu, zenbaki hori eta bere zatitzaileen zerrenda aurkeztu eta prozesu osoa behin eta berriz burutuko da, erabiltzaileak bukatzea aukeratu arte.

Erabiltzaileari zenbakiren baten zatitzaileak nahi al dituen galdetzeko, galdera burutu, erabiltzailearen erantzuna jaso eta erantzuna b edo e izan al den ziurtatzeaz arduratuko den **zat_nahi** izeneko funtzioa definitu beharko da. Erabiltzaileak tekleatutakoa b edo e ez bada, erantzuna egokia ez dela esanez mezu bat aurkeztu eta berriro galdetu beharko da, b edo e lortu arte.

Era berean, erabiltzaileak tekleatutako zenbakia positiboa ez bada, zenbakia ez dela egokia esanez mezu bat aurkeztu eta beste zenbaki bat eskatu beharko da, positiboa den zenbaki bat lortu arte. Eskatze-prozesu hori burutzeko, 2. ariketan definitutako **positiboa_eskatu** izeneko funtzioa erabili beharko da. Funtzio hori zenbaki positibo bat lortu arte zenbakia eskatzeko prozesua errepikatzeaz arduratuko da. Funtzio horrek zenbaki positibo bat eskuratzen duenean, zenbaki hori itzuli beharko du beste funtzio batetik jaso ahal izateko moduan.

Exekuzio-adibidea:

```
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
b
Positiboa den zenbaki oso bat idatzi:
14
14 zenbakiaren zatitzaileen zerrenda: [1, 2, 7, 14]
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
b
Positiboa den zenbaki oso bat idatzi:
12
12 zenbakiaren zatitzaileen zerrenda: [1, 2, 3, 6, 12]
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
e
Exekuzioaren bukaera.
```

Exekuzio-adibidea:

```
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
d
Erantzuna ez da egokia...
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
a
Erantzuna ez da egokia...
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
b
Positiboa den zenbaki oso bat idatzi:
-5
Idatzitako zenbakia ez da positiboa.
Positiboa den zenbaki oso bat idatzi:
0
Idatzitako zenbakia ez da positiboa.
Positiboa den zenbaki oso bat idatzi:
12
12 zenbakiaren zatitzaileen zerrenda: [1, 2, 3, 6, 12]
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
j
Erantzuna ez da egokia...
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
a
Erantzuna ez da egokia...
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
b
Positiboa den zenbaki oso bat idatzi:
7
7 zenbakiaren zatitzaileen zerrenda: [1, 7]
Zenbakiren baten zatitzaileen zerrenda nahi al duzu? (b/e):
e
Exekuzioaren bukaera.
```

Zenbaki positibo baten zatitzaileen zerrenda kalkulatzeko, aukera bat Zerrenda_eraketa.hs moduluan definitutako zatizer_ze funtzioa erabiltzea da eta beste aukera bat zerrenda-eraketaren teknika erabiliz zuzenean zatitzaileen zerrenda sortzea izango litzateke. Enuntziatuan zehaztu den bezala, 2. ariketan definitutako positiboa_eskatu funtzioa ere erabili behar da.