5. GAIKO ARIKETAK
DMA-EN EKUAZIO BIDEZKO ESPEZIFIKAZIOA

A) 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

Jarraian agertzen diren kasuak urratsez urrats **garatu**:
lehen_bik_pos([5, 9, 8, 6])
lehen_bik_pos([5, 9])

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

Jarraian agertzen diren kasuak urratsez urrats **garatu**:

azken_pos(8, [8, 6, 8, 5])
azken_pos(8, [3, 2])

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

Jarraian agertzen den kasua urratsez urrats **garatu**:

hand([5, 3, 8, 7, 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 geldituko dira baina lehenengo zerrendako elementuak alderantzizko ordenean geldituko 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(3, [7, 6, 8, 5, 9]) = [5, 9]

Jarraian agertzen den kasua urratsez urrats **garatu**:
ezabatu(3, [7, 6, 8, 5, 9])

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(3, [7, 6, 8, 5, 9]) = [7, 6, 8]

Jarraian agertzen den kasua urratsez urrats **garatu**:
hartu(3, [7, 6, 8, 5, 9])

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

28) kolapsatu

a) Zenbaki osoz osatutako zerrenda bat emanda, jarraian errepikatuta agertzen diren elementuen kopia bakarra mantenduz osatzen den zerrenda itzultzen duen *kolapsatu* izeneko funtzioaren **espezifikazio ekuazionala** eman. 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]

b) Ekuazioak eman ondoren, kolapsatu([3, 9, 9, 3]) adibidea garatu urrats bakoitzean zein ekuazio erabili den zehaztuz.

29) hedatu

a) 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 *hedatu* izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuliko da. (*hutsa_da*, *leh* eta *hond* funtzioak erabili):

Adibidea:

hedatu([4, 5, 2, 10, 8, 3, 6]) =
= [<u>4</u>, <u>5 + 4</u>, <u>2</u>, <u>10 + 2</u>, <u>8 + 12</u>, <u>3 + 20</u>, <u>6</u>]
= [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

b) Ekuazioak eman ondoren, hedatu([8, 7, 2]) adibidea garatu urrats bakoitzean zein ekuazio erabili den zehaztuz.

30) zapaldu

a) Zenbaki osoz osatutako zerrenda bat emanda, ezkerretik hasi eta elementu bakoitzak bere atzetik datozen zenbaki txikiagoak (handiagoa den bat agertu arte) zapalduz (ordezkatuz) lortzen den zerrenda itzultzen duen *zapaldu* izeneko funtzioaren **espezifikazio ekuazionala** eman. Handiagoa 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]

b) Ekuazioak eman ondoren, zapaldu([3, 9, 9, 3]) adibidea garatu urrats bakoitzean zein ekuazio erabili den zehaztuz.

31) hondoratu

a) 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 funtzioaren **espezifikazio ekuazionala** eman. 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 (*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.

b) Ekuazioak eman ondoren, jarraian dagoen adibidea garatu urrats bakoitzean zein ekuazio erabili den zehaztuz:

hondoratu([**8**, 3, 2, 9])

32) elem_kendu

a) 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 funtzioaren espezifikazio ekuazionala 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, **<u>4</u>**, 8, 3], 3) = [5, 4, 1, 2, 8, 3]

2. adibidea:

elem_kendu([1, 7, **<u>5</u>**, 8], 2) = [1, 7, 8]

b) Ekuazioak eman ondoren, jarraian dagoen adibidea garatu urrats bakoitzean zein ekuazio erabili den zehaztuz:

elem_kendu ([1, 7, **<u>5</u>**, 8], 2)

33) garbitu (2008ko apirila #1)

a) 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 <u>garbitu</u> izeneko funtzioaren **espezifikazio ekuazionala** 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 gelditυz.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

garbitu([5, 8, 4, 1, 9])

34) zeharkatu (2008ko apirila #2)

a) Zenbaki osoz osatutako zerrenda bat emanda, zerrendako lehenengo elementua behean zehazten dena bete arte eskuinera desplazatuz lortzen den zerrenda itzultzen duen zeharkatu izeneko funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen diren **adibideak garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

zeharkatu([5, 8, 4])

zeharkatu([5, 5, 4])

35) zatitu (2008ko ekaina)

a) Zenbaki osoz osatutako zerrenda bat emanda, sarreraro zerrendak baino elementu bat gutxiago duen eta jarraian aipatzen diren bi irizpideak jarraituz lortzen den zerrenda itzultzen duen zatitu izeneko funtzioaren **espezifikazio ekuazionala** 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. Hasierako zerrenda hutsa ez bada baina zerrendako lehenengo elementua 0 baldin bada, errore-mezua aurkeztu beharko da. Hasierako zerrendak elementu bakarra badu (elementu hori zeroren desberdina izanda), 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([**3**, 15]) = [5]
zatitu([**5**, 8, 8, 15, 9]) = [8, 8, 3, 9]	zatitu([**3**, 14]) = [14]

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

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

zatitu([5, 20, 6, 15])

36) superbik (2008ko iraila)

a) Zenbaki osoz osatutako zerrenda bat emanda, zerrendako zenbaki bikoiti denen batura bukaeran ipiniz eta zenbaki bakoitiak mantenduz lortzen den zerrenda itzultzen duen <u>superbik</u> izeneko funtzioaren **espezifikazio ekuazionala** 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 geldituko da.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

superbik([3, 10, 8, 9])

37) metatu (2009ko apirila #1)

a) Zenbaki osoz osatutako zerrenda bat emanda, posizio bakoitzean sarrerako zerrendako lehenengo posiziotik posizio horretara arteko batura duen zerrenda itzultzen duen "metatu" izeneko funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

metatu([10, 8, 15])

38) aurreratu (2009ko apirila #2)

a) 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 zerrenda posizio horretan dagoen elementuak aurrera egingo du hurrengo posiziokoarekin lekuz trukatuz eta zerrendak zeharkatzen jarraituko da.
- Lehenengo zerrendako posizioan False dagoen bakoitzean bigarren zerrendan posizio horretan dagoen elementuak ez du aurrera egingo 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]

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

aurreratu([**True**, False, **True**, **True**, **True**], [8, 3, 9, 5, 2])

39) kendubikpos (2009ko ekaina)

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 funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

kendubikpos([8, 5, 16, 7, 10, 4], 3)

40) azk_lehena (2009ko iraila)

a) Zenbaki osoz osatutako zerrenda bat emanda, zerrendako azkeneko zenbaki lehena (eskuinerago dagoena) itzultzen duen *azk_lehena* izeneko funtzioaren **espezifikazio ekuazionala** eman. Emandako zerrenda hutsa baldin bada edo zerrenda hutsa ez izanda zerrendan zenbaki lehenik ez badago, funtzioak -1 itzuli beharko du.

Ekuazioak ematerakoan zerrendako lehenengo elementua itzultzen duen *leh* funtzioa, zerrendako lehenengo elementua kenduz gelditzen den zerrenda, hau da, hondarra itzultzen duen *hond* funtzioa eta zerrenda hutsa al den ala ez esaten duen *hutsa_da* funtzioa erabili.

Gainera zenbaki oso bat emanda zenbakia lehena bada *true* eta bestela *false* itzultzen duen *lehena_da* izeneko funtzio laguntzailea erabili behar da. Funtzio laguntzaile hori definituta dagoela suposatu behar da, ez da definitu behar, erabili egin behar da bakarrik. Gogoratu zenbaki oso bat lehena dela positiboa bada eta justu bi zatitzaile bakarrik baldin baditu.

Adibideak:

- azk_lehena([8, 11, -7, 9, 3, 6]) = 3

  Kasu honetan zerrendan bi zenbaki lehen daude, 11 eta 3, eta funtzioak azkena itzuli behar du, hau da, 3.

- azk_lehena([8, -7, 16, 4, 10]) = -1

  Kasu honetan zerrendan ez dago zenbaki lehenik eta horregatik emaitza -1 da.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

azk_lehena([8, 11, -7, 9, 3, 6])

41) azpiluzbikgehi (2010eko apirila #1)

a) 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 funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

azpiluzbikgehi([10, 10, 10, 8, 8, 15, 8])

42) handibik (2010eko apirila #2)

a) Zenbaki osoz osatutako zerrenda bat emanda, bikote bakoitzeko elementu handiena bi aldiz ipiniz eta txikiena ezabatuz lortzen den zerrenda itzultzen duen "handibik" izeneko funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

handibik([10, 8, 5, 7, 7, 20, 5, 5])

43) kokatu (2010eko ekaina)

a) 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 funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

kokatu([8, 0, 0, 7], [3, 20, 12, 28])

44) azpiluzbikken (2010eko iraila)

a) 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 funtzioaren **espezifikazio ekuazionala** 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.

b) Ekuazioak eman eta gero, jarraian agertzen den **adibidea garatu** urrats bakoitzean zein ekuazio erabili den zehaztuz:

azpiluzbikken([10, 10, 10, 8, 8, 15, 8])

B) <u>Indukzio bidezko frogak zerrendentzat</u>

Jarraian aipatzen diren teorema induktiboak frogatu:

1) luzera(s) = luzera(alder(s))

Edozein zerrenda (s) hartuta, honako berdintza hau betetzen dela frogatu indukzioa erabiliz:

luzera(s) = luzera(alder(s))

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = x:r izango da. <u>Indukzio-hipotesia</u> r zerrendak propietatea bete egiten duela suposatzea izango da. Hor *alder* funtzioaren 7.a bertsioa kontsideratu (++ erabiliz). Gainera edozein u eta v bi zerrenda hartuta, honako propietate hau betetzen dela kontuan hartu beharko da: *luzera(u ++ v) = luzera(u) + luzera(v)*

2) luzera(s) ≥ luzera(kendu(x, s))

Edozein elementu (x) eta edozein zerrenda (s) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

luzera(s) ≥ luzera(kendu(x, s))

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = z:r izango da. <u>Indukzio-hipotesia</u> r zerrendak propietatea bete egiten duela suposatzea izango da.

3) s ++ [ ] = s

Edozein zerrenda hartuta (s) honako hau betetzen dela frogatu indukzioa erabiliz:

s ++ [] = s

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = x:r izango da. <u>Indukzio-hipotesia</u> r zerrendak propietatea bete egiten duela suposatzea izango da.

4) alder(s ++ r) = alder(r) ++ alder(s)

Edozein bi zerrenda hartuta (s eta r) honako hau betetzen dela frogatu indukzioa erabiliz:

alder(s ++ r) = alder(r) ++ alder(s)

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = x:w izango da. <u>Indukzio-hipotesia</u> w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

<u>Oinarrizko kasuan</u> propietatea bete egiten dela frogatzeko, 3. ariketan frogatutako propietatea erabili beharko da (s zerrenda bat hartuta, honakoa beteko da s ++ [] = s).

5) aldiz(z, s ++ r) = aldiz(z, s) + aldiz(z, r)

Edozein elementu (z) eta edozein bi zerrenda (s eta r) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

aldiz(z, s ++ r) = aldiz(z, s) + aldiz(z, r)

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = x:w izango da. Indukzio-hipotesia w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

6) aldiz(x, kendu(x, s)) = 0

Edozein elementu (x) eta edozein zerrenda (s) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

aldiz(x, kendu(x, s)) = 0

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = z:r izango da. Indukzio-hipotesia x elementuak eta r zerrendak propietatea bete egiten dutela suposatzea izango da.

7) kendu(x, s ++ r) = kendu(x, s) ++ kendu(x, r)

Edozein elementu (x) eta edozein bi zerrenda (s eta r) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

kendu(x, s ++ r) = kendu(x, s) ++ kendu(x, r)

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = z:w izango da. Indukzio-hipotesia x elementuak eta w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

8) luzera(kendu(x, s ++ r)) = luzera(kendu(x, s)) + luzera(kendu(x, r))

t motakoa den edozein elementu (x) eta [t] motakoak diren edozein bi zerrenda (s eta r) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

luzera(kendu(x, s ++ r)) = luzera(kendu(x, s)) + luzera(kendu(x, r))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = z:w izango da. Indukzio-hipotesia x elementuak eta w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

9) kendu(x, kendu(x, s)) ++ r = kendu(x, s) ++ r

t motakoa den edozein elementu (x) eta [t] motakoak diren edozein bi zerrenda (s eta r) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

kendu(x, kendu(x, s)) ++ r = kendu(x, s) ++ r

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = z:w izango da. Indukzio-hipotesia x elementuak eta w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

10) batu(inkr(s)) = batu(s) + luzera(s)

[t] motakoa den edozein zerrenda (s) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

batu(inkr(s)) = batu(s) + luzera(s)

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = x:r izango da. Indukzio-hipotesia r zerrendak propietatea bete egiten duela suposatzea izango da.

11) bik_kendu(s ++ r) = bik_kendu(s) ++ bik_kendu(r)

Edozein bi zerrenda (s eta r) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

bik_kendu(s ++ r) = bik_kendu(s) ++ bik_kendu(r)

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = x:w izango da. Indukzio-hipotesia w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

12) luzera(pos_bik_kendu(s)) = luzera(s) `div` 2, s-ren luzera bikoitia denean

Luzera bikoitia duen edozein zerrenda (s) hartuta, honako propietate hau betetzen dela frogatu:

luzera(pos_bik_kendu(s)) = luzera(s) `div` 2

`div` zatiketa osoa izanda.

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = x:r izango da baina s-ren elementu-kopurua bikoitia dela gogoratu beharko dugu eta ondorioz r ez dela hutsa izango ere badakigu (gutxienez elementu bat du). <u>Indukzio-hipotesia</u> hond(r) zerrendak propietatea bete egiten duela suposatzea izango da. Konturatu hond(r) zerrendaren luzera bikoitia izango dela. Hutsa ez den edozein r zerrenda hartuta honako propietate hau beteko dela kontuan hartu behar da:

luzera(r) = 1 + luzera(hond(r))

13) luzera(pos_bik_kendu(s)) = (luzera(s) `div` 2) + 1, s-ren luzera bakoitia denean

Luzera bakoitia duen edozein zerrenda (s) hartuta, honako propietate hau betetzen dela frogatu:

luzera(pos_bik_kendu(s)) = (luzera(s) `div` 2) + 1

`div` zatiketa osoa izanda.

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = x:[] izango da eta <u>kasu orokorra</u> s = x:(y:r) izango da baina s-ren elementu-kopurua bakoitia dela gogoratu beharko dugu eta ondorioz r ez dela hutsa izango ere badakigu (gutxienez elementu bat du). <u>Indukzio-hipotesia</u> r zerrendak propietatea bete egiten duela suposatzea izango da. Konturatu r zerrendaren luzera bakoitia izango dela.

14) pos_bik_kendu*(s ++ r)* = pos_bik_kendu*(s)* ++ pos_bik_kendu*(r), s-ren luzera bikoitia denean*

<u>Luzera bikoitia</u> duen zerrenda bat (s) eta beste edozein zerrenda (r) hartuta, honako hau betetzen dela frogatu indukzioa erabiliz:

pos_bik_kendu(s ++ r) = pos_bik_kendu(s) ++ pos_bik_kendu(r)

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = x:(y:w) izango da baina s-ren elementu-kopurua bikoitia dela gogoratu beharko dugu eta ondorioz w-ren luzera ere bikoitia izango da. <u>Indukzio-hipotesia</u> w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

15) pos_bik_kendu*(s ++ r)* ≠ pos_bik_kendu*(s)* ++ pos_bik_kendu*(r), s-ren luzera bakoitia denean*

s zerrendaren luzera bikoitia ez denean aurreko ariketako propietatea ez dela betetzen egiaztatu. Horretarako jarraian datozen espresioak kalkulatu (ez da beharrezkoa kalkulua urratsez urrats egitea):

- *pos_bik_kendu*([7, 4, 8] ++ [5, 2])
- *pos_bik_kendu*([7, 4, 8]) ++ *pos_bik_kendu*([5, 2])

Oharra: Zerrenda denek propietate bat betetzen dutela frogatzeko, indukzioa bezalako teknikaren bat erabili behar da, baina propietate bat betetzen ez duten zerrendak badaudela frogatzeko nahikoa da propietatea ez dela frogatzen erakusten duen adibide bat ematearekin.

16)aldiz(x, s) = aldiz(x + 1, inkr(s)) (2008ko apirila #1)

a) t motako elementu bat eta t motako zerrenda bat emanda, elementu hori zerrenda horretan zenbat aldiz agertzen den kalkulatzen duen aldiz izeneko funtzioaren **espezifikazio ekuazionala** eman:

aldiz:: (t, [t]) → Int

Adibideak:

aldiz(5, [2, 6]) = 0
aldiz(8, [3, 4, 8, 9, 8]) = 2

b) Zenbaki osoz osatutako zerrenda bat emanda, zerrendako elementu bakoitzari 1 batuz lortzen diren elementuez osatutako zerrenda itzultzen duen inkr izeneko funtzioaren **espezifikazio ekuazionala** eman. Zerrenda hutsa bada, zerrenda hutsa itzuli beharko da emaitza bezala:

inkr:: ([t]) → [t]

Adibideak:

inkr([2, 6]) = [3, 7]
inkr([3, 4, 8, 9, 8]) = [4, 5, 9, 10, 9]

c) Edozein x elementu eta edozein s zerrenda hartuta (x elementua s zerrendako elementuen mota berekoa dela kontsideratuz), honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

aldiz(x, s) = aldiz(x + 1, inkr(s))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = z:r izango da. Indukzio-hipotesia x elementuak eta r zerrendak propietatea bete egiten dutela suposatzea izango da.

17)batu(s) = azkena(s) + batu(azkena_kendu(s)) (2008ko apirila #2)

a) Int motako zerrenda bat emanda, zerrendako elementu denen batura itzultzen duen batu izeneko funtzioaren **espezifikazio ekuazionala** eman**.**

batu:: ([Int]) $\rightarrow$ Int

Adibideak:

batu([5, -1, 8]) = 12
batu([8, 3, 7, 8]) = 26

b) Int motako zerrenda bat emanda, azkeneko elementua (eskuineko ertzean dagoena) itzultzen duen azkena izeneko funtzioaren **espezifikazio ekuazionala** eman. Zerrenda hutsa bada, errore-mezua itzuliko du. Ekuazioak ematerakoan *hutsa_da* funtzioa erabili.

azkena:: ([t]) $\rightarrow$ t

Adibideak:

azkena([5, 2, 6]) = 6
azkena([3, 8, 4, 4, 9, 8]) = 8

c) Int motako zerrenda bat emanda, azkeneko elementua (eskuineko ertzean dagoena) kenduz gelditzen den zerrenda itzultzen duen *azkena_kendu* izeneko funtzioaren **espezifikazio ekuazionala** eman. Sarrerako zerrenda hutsa bada, errore-mezua itzuliko du. Ekuazioak ematerakoan *hutsa_da* funtzioa erabili.

azkena_kendu:: ([t]) $\rightarrow$ [t]

Adibideak:

azkena_kendu([5, 2, 6]) = [5, 2]
azkena_kendu([3, 8, 4, 4, 9, 8]) = [3, 8, 4, 4, 9]

d) Hutsa ez den Int motako edozein s zerrenda hartuta, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

batu(s) = azkena(s) + batu(azkena_kendu(s))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = x:[] izango da eta kasu orokorra s = z:r izango da, baina r hutsa ez dela jakinda. Indukzio-hipotesia r zerrendak propietatea bete egiten duela suposatzea izango da.

18)badago(x, s) = badago(x + 1, inkr(s)) (2008ko ekaina)

a) t motako elementu bat eta t motako zerrenda bat emanda, elementua zerrendan badago True eta bestela False itzultzen duen badago izeneko funtzioaren espezifikazio ekuazionala eman:

badago:: (t, [t]) → Bool

Adibideak:

badago(5, [2, 6]) = False	badago(8, [3, 4, 8, 9, 8]) = True

b) Zenbaki osoz osatutako zerrenda bat emanda, zerrendako elementu bakoitzari 1 batuz lortzen diren elementuez osatutako zerrenda itzultzen duen inkr izeneko funtzioaren **espezifikazio ekuazionala** eman. Zerrenda hutsa bada, zerrenda hutsa itzuli beharko da emaitza bezala:

inkr:: ([t]) → [t]

Adibideak:

inkr([2, 6]) = [3, 7]	inkr([3, 4, 8, 9, 8]) = [4, 5, 9, 10, 9]

c) Edozein x elementu eta edozein s zerrenda hartuta (x elementua s zerrendako elementuen mota berekoa dela kontsideratuz), honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

badago(x, s) = badago(x + 1, inkr(s))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = z:r izango da. Indukzio-hipotesia x elementuak eta r zerrendak propietatea bete egiten dutela suposatzea izango da.

19)luzera(kendu(e, s)) = luzera(s) – aldiz(e, s) (2008ko iraila)

a) t motako zerrenda bat emanda, zerrendan zenbat elementu dauden kalkulatzen duen <u>luzera</u> izeneko funtzioaren **espezifikazio ekuazionala** eman:

luzera:: ([t]) $\rightarrow$ Int

Adibideak:

luzera([d5, d2, d6]) = 3	luzera([d8, d3, d4, d8, d9, d8]) = 6

b) t motako elementu bat eta t motako zerrenda bat emanda, elementu horren agerpen denak kenduz lortzen den zerrenda itzultzen duen <u>kendu</u> izeneko funtzioaren **espezifikazio ekuazionala** eman:

kendu:: (t, [t]) $\rightarrow$ [t]

Adibideak:

kendu(d8, [d5, d2, d6]) = [d5, d2, d6]	kendu(d8, [**d8**, d3, d4, **d8**, d9, **d8**]) = [d3, d4, d9]

c) t motako elementu bat eta t motako zerrenda bat emanda, elementu hori zerrendan zenbat aldiz agertzen den kalkulatzen duen <u>aldiz</u> izeneko funtzioaren **espezifikazio ekuazionala** eman:

aldiz:: (t, [t]) $\rightarrow$ Int

Adibideak:

aldiz(d5, [d2, d6]) = 0	aldiz(d8, [d3, d4, d8, d9, d8]) = 2

d) Edozein e elementu eta <u>hutsa ez den</u> edozein s zerrenda hartuta (e elementua s zerrendako elementuen mota berekoa dela kontsideratuz), honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

luzera(kendu(e, s)) = luzera(s) – aldiz(e, s)

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> s = [] izango da eta <u>kasu orokorra</u> s = x:r izango da. <u>Indukzio-hipotesia</u> e elementuak eta r zerrendak propietatea bete egiten dutela suposatzea izango da.

20)alder(alder(s)) = s (2009ko apirila #1)

a) t motako bi zerrenda emanda, zerrenda biak elkartuz lortzen den zerrenda itzultzen duen ++ eragilearen **espezifikazio ekuazionala** eman:

++:: ([t], [t]) → [t]

Adibideak: [1, 7] ++ [8, 5, 9] = [1, 7, 8, 5, 9]
[] ++ [8, 5, 9] = [8, 5, 9]

b) t motako zerrenda bat emanda, elementuak alderantzizko ordenean ipiniz lortzen den zerrenda itzultzen duen "alder" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli beharko da. Ekuazioak ematerakoan bi zerrenda elkartzeko balio duen ++ eragilea erabili:

alder:: ([t]) → [t]

Adibideak: alder([2, 6]) = [6, 2]
alder([3, 4, 8, 9, 8]) = [8, 9, 8, 4, 3]

c) Edozein s zerrenda hartuta, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

alder(alder(s)) = s

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = x:r izango da. Indukzio-hipotesia r zerrendak propietatea bete egiten duela suposatzea izango da. Gainera zerrenden gaineko honako propietate hau erabili beharko da:

- Edozein bi zerrenda v eta w hartuta, honako hau betetzen da

alder(v ++ w) = alder(w) ++ alder(v) **(Prop)**

21) *luzera(hond(s)) = luzera(azkena_kendu(s)) (2009ko apirila #2)*

a) t motako zerrenda bat emanda, zerrendan zenbat elementu dauden kalkulatzen duen "*luzera*" funtzioaren **espezifikazio ekuazionala** eman. Zerrenda hutsa bada 0 itzuliko du:

luzera :: ([t]) → Int

Adibidea: luzera[9, 7, 8, 8, 1] = 5

b) t motako zerrenda bat emanda, zerrendako azkeneko elementua kenduz gelditzen den zerrenda itzultzen duen "*azkena_kendu*" funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada errore-mezua aurkeztuko du. Hasierako zerrendak elementu bakarra badu, zerrenda hutsa itzuliko du.

azkena_kendu :: ([t]) → [t]

Adibidea: azkena_kendu([8, 6, 7, 3]) = [8, 6, 7]

c) t motako zerrenda bat emanda, zerrendako lehenengo elementua (ezkerreko ertzekoa) kenduz gelditzen den zerrenda, hau da, hondarra itzultzen duen "hond" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, errore-mezua aurkeztuko du:

hond :: ([t]) → [t]
Adibidea: hond([8, 6, 7, 3]) = [6, 7, 3]

d) Hutsa ez den edozein s zerrenda hartuta, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

luzera(hond(s)) = luzera(azkena_kendu(s))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua **s = z:[]** izango da eta kasu orokorra **s = x:r** izango da, **r hutsa ez den zerrenda bat izanik**. Indukzio-hipotesia hutsa ez den **r zerrendak propietatea bete egiten du**ela suposatzea izango da. Gainera zerrenden gaineko honako propietate hau erabili beharko da:

- Hutsa ez den edozein w zerrenda hartuta, honako hau betetzen da

luzera(w) = 1 + luzera(hond(w)) **(Prop)**

22)aldiz(x, s) ≥ aldiz(x, azkena_kendu(s)) (2009ko ekaina)

a) t motako elementu bat eta t motako zerrenda bat emanda, elementua zerrendan zenbat aldiz agertzen den kalkulatzen duen "*aldiz*" izeneko funtzioaren espezifikazio ekuazionala eman:

aldiz:: (t, [t]) → Int

Adibideak: aldiz(5, [8, 5, 6, 5]) = 2
aldiz(8, []) = 0
aldiz(8, [7, 1, 2]) = 0

b) t motako zerrenda bat emanda, zerrendako azkeneko elementua kenduz gelditzen den zerrenda itzultzen duen "*azkena_kendu*" funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada errore-mezua aurkeztuko du. Hasierako zerrendak elementu bakarra badu, zerrenda hutsa itzuliko du.

azkena_kendu:: ([t]) → [t]

Adibidea: azkena_kendu([8, 6, 7, 3]) = [8, 6, 7]

c) Edozein x elementu eta hutsa ez den edozein s zerrenda hartuta (x elementua s zerrendako elementuen mota berekoa dela kontsideratuz), honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

$$aldiz(x, s) \geq aldiz(x, azkena\_kendu(s))$$

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> **s = z:[]** izango da eta <u>kasu orokorra</u> s = z:r izango da, r hutsa ez den zerrenda bat izanda. <u>Indukzio-hipotesia</u> x elementuak eta hutsa ez den r zerrendak propietatea bete egiten dutela suposatzea izango da.

23) *hond(alder(s)) = alder(azkena_kendu(s)) (2009ko iraila)*

a) t motako bi zerrenda emanda, zerrenda biak elkartuz lortzen den zerrenda itzultzen duen ++ eragilearen **espezifikazio ekuazionala** eman:

++:: ([t], [t]) → [t]
Adibideak: [1, 7] ++ [8, 5, 9] = [1, 7, 8, 5, 9]
[] ++ [8, 5, 9] = [8, 5, 9]

b) t motako zerrenda bat emanda, elementuak alderantzizko ordenean ipiniz lortzen den zerrenda itzultzen duen "<u>alder</u>" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli beharko da. Ekuazioak ematerakoan bi zerrenda elkartzeko balio duen ++ eragilea erabili:

alder:: ([t]) → [t]
Adibideak: alder([2, 6]) = [6, 2]
alder([3, 4, 8, 9, 8]) = [8, 9, 8, 4, 3]

c) t motako zerrenda bat emanda, zerrendako lehenengo elementua (ezkerreko ertzekoa) kenduz gelditzen den zerrenda, hau da, hondarra itzultzen duen "<u>hond</u>" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, errore-mezua aurkeztuko du:

hond :: ([t]) → [t]
Adibidea: hond([8, 6, 7, 3]) = [6, 7, 3]

d) t motako zerrenda bat emanda, zerrendako azkeneko elementua kenduz gelditzen den zerrenda itzultzen duen "*azkena_kendu*" funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada errore-mezua aurkeztuko du. Hasierako zerrendak elementu bakarra badu, zerrenda hutsa itzuliko du.

azkena_kendu :: ([t]) → [t]
Adibidea: azkena_kendu([8, 6, 7, 3]) = [8, 6, 7]

e) Hutsa ez den edozein s zerrenda hartuta, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

hond(alder(s)) = alder(azkena_kendu(s))

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> **s = z:[]** izango da eta <u>kasu orokorra</u> s = z:r izango da, r hutsa ez den zerrenda bat izanda. <u>Indukzio-hipotesia</u> hutsa ez den r zerrendak propietatea bete egiten duela suposatzea izango da. Gainera zerrenden gaineko honako <u>propietate hauek erabili</u> beharko da:

- Hutsa ez den edozein w zerrenda hartuta eta edozein v zerrenda hartuta, honako hau betetzen da:
  hond(w ++ v) = hond(w) ++ v **(Prop1)**
- Edozein w zerrenda hartuta, honako hau betetzen da:
  luzera(w) = luzera(alder(w)) **(Prop2)**

24)batu(s) = batu(alder(s)) (2010eko apirila #1)

a) t motako bi zerrenda emanda, zerrenda biak elkartuz lortzen den zerrenda itzultzen duen ++ eragilearen **espezifikazio ekuazionala** eman:

++:: ([t], [t]) → [t]

Adibideak:
[1, 7] ++ [8, 5, 9] = [1, 7, 8, 5, 9]
[] ++ [8, 5, 9] = [8, 5, 9]

b) Osoak diren zenbakiz osatutako zerrenda bat emanda, zerrendako elementu denen batura itzultzen duen "batu" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa baldin bada, funtzioak 0 balioa itzuli beharko du emaitza bezala:

batu:: ([Int]) → Int

Adibideak: batu([4, 6, 5]) = 15

c) t motako zerrenda bat emanda, elementuak alderantzizko ordenean ipiniz lortzen den zerrenda itzultzen duen "alder" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli beharko da. Ekuazioak ematerakoan bi zerrenda elkartzeko balio duen ++ eragilea erabili:

alder:: ([t]) → [t]

Adibideak:
alder([2, 6]) = [6, 2]
alder([3, 4, 8, 9, 8]) = [8, 9, 8, 4, 3]

d) Zenbaki osoz osatutako edozein s zerrenda hartuta, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

batu(s) = batu(alder(s))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = x:r izango da. Indukzio-hipotesia r zerrendak propietatea bete egiten duela suposatzea izango da. Gainera zerrenden gaineko honako propietate hau erabili beharko da:

- Int motako edozein bi zerrenda v eta w hartuta, honako hau betetzen da:

batu(v ++ w) = batu(v) + batu(w) (Prop)

25)inkr(alder(s)) = alder(inkr(s)) (2010eko apirila #2)

a) t motako bi zerrenda emanda, zerrenda biak elkartuz lortzen den zerrenda itzultzen duen ++ eragilearen **espezifikazio ekuazionala** eman:

++:: ([t], [t]) → [t]

Adibideak:

[1, 7] ++ [8, 5, 9] = [1, 7, 8, 5, 9]
[] ++ [8, 5, 9] = [8, 5, 9]

b) Osoak diren zenbakiz osatutako zerrenda bat emanda, zerrendako elementu bakoitzari 1 gehituz (inkrementatuz) lortzen den zerrenda itzultzen duen "inkr" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa baldin bada, funtzioak zerrenda hutsa itzuli beharko du emaitza bezala:

inkr:: ([Int]) → [Int]

Adibideak:

inkr([4, 9, 2]) = [5, 10, 3]

c) t motako zerrenda bat emanda, elementuak alderantzizko ordenean ipiniz lortzen den zerrenda itzultzen duen "alder" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli beharko da. Ekuazioak ematerakoan bi zerrenda elkartzeko balio duen ++ eragilea erabili:

alder:: ([t]) → [t]

Adibideak:

alder([2, 6]) = [6, 2]
alder([3, 4, 8, 9, 8]) = [8, 9, 8, 4, 3]

d) Zenbaki osoz osatutako edozein s zerrenda hartuta, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz**:**

inkr(alder(s)) = alder(inkr(s))

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua s = [] izango da eta kasu orokorra s = x:r izango da. Indukzio-hipotesia r zerrendak propietatea bete egiten duela suposatzea izango da. Gainera zerrenden gaineko honako propietate hau erabili beharko da:

- Edozein bi zerrenda v eta w hartuta, honako hau betetzen da:

inkr(v ++ w) = inkr(v) ++ inkr(w) **(Prop)**

26)aldiz(x, alder(s)) = aldiz(x, s) (2010eko ekaina)

a) t motako bi zerrenda emanda, zerrenda biak elkartuz lortzen den zerrenda itzultzen duen ++ eragilearen **espezifikazio ekuazionala** eman:

++:: ([t], [t]) → [t]

Adibideak:

[1, 7] ++ [8, 5, 9] = [1, 7, 8, 5, 9]
[] ++ [8, 5, 9] = [8, 5, 9]

b) t motako elementu bat eta t motako zerrenda bat emanda, elementua zerrendan zenbat aldiz agertzen den kalkulatzen duen "*aldiz*" izeneko funtzioaren espezifikazio ekuazionala eman:

aldiz :: (t, [t]) → Int

Adibideak:

aldiz(d5, [d8, d5, d6, d5]) = 2
aldiz(d8, []) = 0
aldiz(d8, [d7, d1, d2]) = 0

c) t motako zerrenda bat emanda, elementuak alderantzizko ordenean ipiniz lortzen den zerrenda itzultzen duen "alder" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa bada, zerrenda hutsa itzuli beharko da. Ekuazioak ematerakoan bi zerrenda elkartzeko balio duen ++ eragilea erabili.

alder :: ([t]) → [t]

Adibideak:

alder([d2, d6]) = [d6, d2]
alder([d3, d4, d8, d9, d8]) = [d8, d9, d8, d4, d3]

d) Edozein x elementu eta edozein s zerrenda hartuta (x elementua s zerrendako elementuen mota berekoa dela kontsideratuz), honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

aldiz(x, alder(s)) = aldiz(x, s)

Indukzioa s-ren gainean aplikatu behar da. Oinarrizko kasua **s = []** izango da eta kasu orokorra **s = z:r** izango da. Indukzio-hipotesia x elementuak eta r zerrendak propietatea bete egiten dutela suposatzea izango da. Gainera h elementu bat izanda eta v eta w bi zerrenda izanda, jarraian adierazten dena betetzen dela dioen Prop propietatea erabili beharko da:

aldiz(h, v ++ w) = aldiz(h, v) + aldiz(h, w) **(Prop)**

27)inkr(s ++ r) = inkr(s) ++ inkr(r) (2010eko iraila)

a) t motako bi zerrenda emanda, zerrenda biak elkartuz lortzen den zerrenda itzultzen duen ++ eragilearen **espezifikazio ekuazionala** eman:

++:: ([t], [t]) $\rightarrow$ [t]

Adibideak:

[1, 7] ++ [8, 5, 9] = [1, 7, 8, 5, 9]
[] ++ [8, 5, 9] = [8, 5, 9]

b) Osoak diren zenbakiz osatutako zerrenda bat emanda, zerrendako elementu bakoitzari 1 gehituz (<u>inkr</u>ementatuz) lortzen den zerrenda itzultzen duen "inkr" izeneko funtzioaren **espezifikazio ekuazionala** eman. Hasierako zerrenda hutsa baldin bada, funtzioak zerrenda hutsa itzuli beharko du emaitza bezala:

inkr:: ([Int]) $\rightarrow$ [Int]

Adibideak:

inkr([4, 6, 5]) = [5, 7, 6]

c) s eta r edozein bi zerrenda izanda, honako propietate hau betetzen dela frogatu **indukzioa** erabiliz:

inkr(s ++ r) = inkr(s) ++ inkr(r)

Indukzioa s-ren gainean aplikatu behar da. <u>Oinarrizko kasua</u> **s = []** izango da eta <u>kasu orokorra</u> s = z:w izango da. <u>Indukzio-hipotesia</u> w eta r zerrendek propietatea bete egiten dutela suposatzea izango da.

C) <u>Pilen gaineko eragiketak</u>

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

1) irauli

Bi pila emanda, lehenengo pila bigarrenaren gainean irauliz lortzen den pila itzultzen duen funtzioa: *irauli*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)

		t8	
		t14	p1 iraulita
		t5	
t10		t6	
t6		t10	
t5	t5	t5	
t14	t7	t7	p2
t8	t20	t20	
p1	p2	irauli(p1, p2)	

irauli funtzioa era errekurtsiboan definitu behar da beste funtzio laguntzailerik erabili gabe.

2) alder

Pila bat emanda, elementuak alderantzizko ordenean ipiniz lortzen den pila itzultzen duen funtzioa: *alder*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)

t10	t8
t6	t14
t5	t5
t14	t6
t8	t10
p	alder(p)

Ariketa honetan eskatzen den *alder* funtzioa aurreko ariketako *irauli* funtzioa erabiliz definitu behar da. Ondorioz, *alder* funtzioa ez da errekurtsiboa izango.

3) ipini

Bi pila emanda, lehenengo pila bigarrenaren gainean ipiniz lortzen den pila itzultzen duen funtzioa: *ipini*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)

t10
t6
t5
t14
t8

p1

t5
t7
t20

p2

t10	p1
t6	
t5	
t14	
t8	
t5	p2
t7	
t20	

ipini(p1, p2)

D) <u>Indukzio bidezko frogak pilentzat</u>

Jarraian aipatzen diren teorema induktiboak frogatu:

1) altuera(irauli(p, q)) = altuera(p) + altuera(q)

p eta q edozein bi pila izanda, honako hau betetzen dela frogatu indukzioa erabiliz:

altuera(irauli(p, q)) = altuera(p) + altuera(q)

2) altuera(ipini(p, q)) = altuera(p) + altuera(q)

p eta q edozein bi pila izanda, honako hau betetzen dela frogatu indukzioa erabiliz:

altuera(ipini(p, q)) = altuera(p) + altuera(q)

E) <u>Ilaren gaineko eragiketak</u>

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

1) elkartu

t motako bi ilara emanda, ilara biak elkartuz (bigarrena lehenengoaren atzean ipiniz) lortzen den ilara itzultzen duen funtzioa: *elkartu*.

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

elkartu(<< t3, t4 >>, << t5, t6 >>) = << t3, t4, t5, t6 >>
elkartu(<< >>, << t5, t6 >>) = << t5, t6 >>

2) batu

Int motako ilara bat emanda, ilarako elementuen batura itzultzen duen funtzioa: *batu*.

Adibideak:

batu(<< >>) = 0
batu(<< 9, 1, 40 >>) = 50

F) <u>Indukzio bidezko frogak ilarentzat</u>

Jarraian aipatzen diren teorema induktiboak frogatu:

1) batu(elkartu(c, d)) = batu(c) + batu(d)

c eta d edozein bi ilara izanda, honako berdintza hau betetzen dela frogatu indukzioa erabiliz:

batu(elkartu(c, d)) = batu(c) + batu(d)

2) batu(elkartu(c, d)) = batu(elkartu(d, c))

c eta d edozein bi ilara izanda, honako berdintza hau betetzen dela frogatu indukzioa erabiliz:

batu(elkartu(c, d)) = batu(elkartu(d, c))

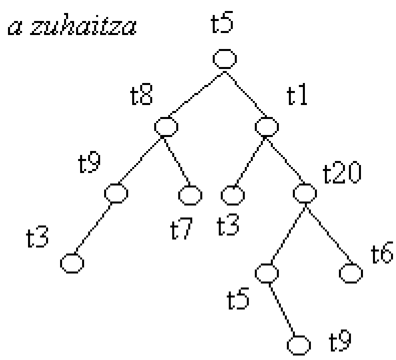
G) Zuhaitz bitarren gaineko eragiketak

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

1) aldiz

Elementu bat zuhaitz bitar batean zenbat aldiz agertzen den kalkulatzen duen funtzioa: *aldiz*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



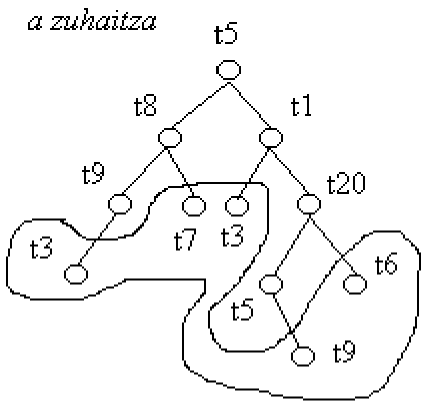
aldiz(t5, *a*) = 2
aldiz(t12, *a*) = 0

aldiz(t8, Zhutsa) = 0

2) hostozer

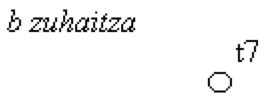
Zuhaitz bitar bateko hostoez osatutako zerrenda itzultzen duen funtzioa: *hostozer*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



hostozer(*a*) = [t3, t7, t3, t9, t6]

hostozer(Zhutsa) = [ ]

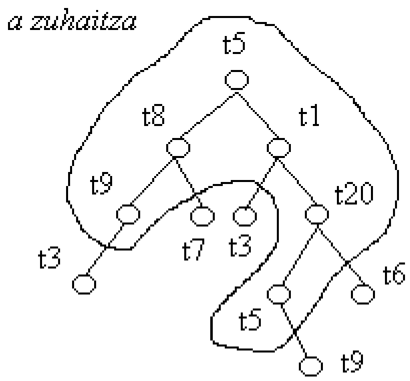


hostozer(*b*) = [t7]

3) barnekop

Zuhaitz bitar bat emanda, hostoak ez diren (barnekoak diren) zenbat adabegi dituen kalkulatzen duen funtzioa: *barnekop*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



barnekop(*a*) = 6

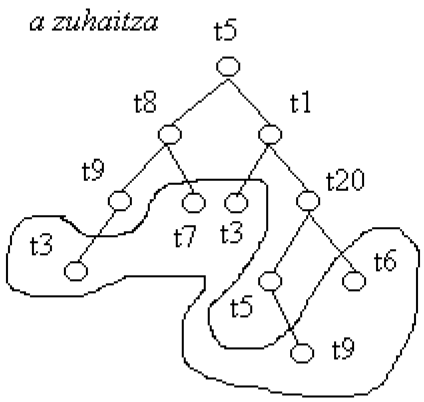
barnekop(Zhutsa) = 0

barnekop(*b*) = 0

4) hostokop

Zuhaitz bitar bat emanda, hosto kopurua kalkulatzen duen funtzioa: *hostokop*

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



hostokop(*a*) = 5

hostokop(Zhutsa) = 0

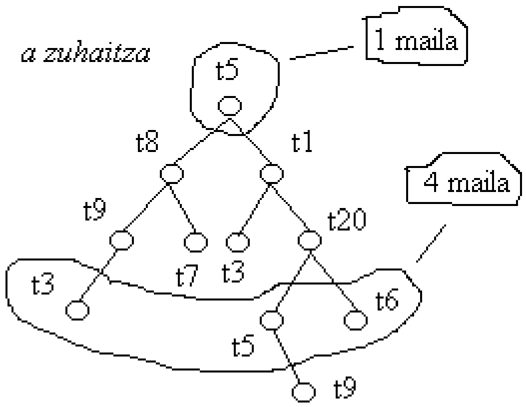
hostokop(*b*) = 1

5) mailakop

Zuhaitz bitar bat eta maila adierazten duen zenbaki oso bat emanda, maila horretako adabegien kopurua itzultzen duen funtzioa: *mailakop*.

Gogoratu erroaren maila 1 dela eta beste edozein adabegirentzat bere maila bere gurasoaren maila gehi bat dela. Maila bezala 0 edo txikiagoa den zenbaki bat emanez gero, errore mezu bat aurkeztu beharko da.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



mailakop(*a*, 1) = 1

mailakop(*a*, 4) = 3

mailakop(*a*, 7) = 0

mailakop(*a*, 0) = error

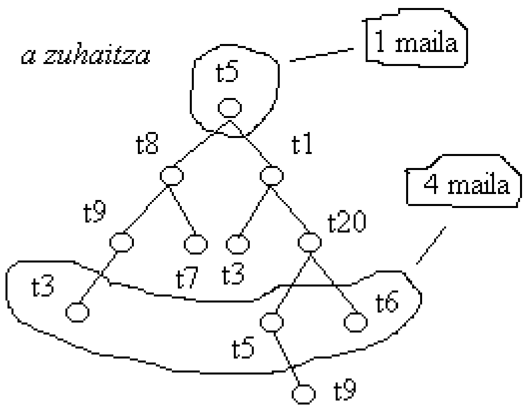
mailakoak(Zhutsa, 2) = 0

6) mailazer

Zuhaitz bitar bat eta maila adierazten duen zenbaki oso bat emanda, maila horretako adabegiez osatutako zerrenda (ezkerretik eskuinera) itzultzen duen funtzioa: *mailazer*.

Maila bezala 0 edo txikiagoa den zenbaki bat emanez gero, errore mezu bat aurkeztu beharko da.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



mailazer(*a*, 1) = [t5]

mailazer(*a*, 4) = [t3, t5, t6]

mailazer(*a*, 7) = [ ]

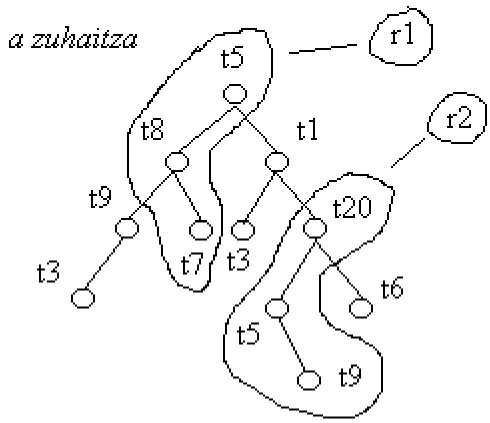
mailazer(*a*, 0) = error

mailazer(Zhutsa, 2) = [ ]
mailazer(Zhutsa, 0) = error

7) adarra_da

Zerrenda bat eta zuhaitz bitar bat emanda, zerrenda hori zuhaitzeko adar batekin bat al datorren erabakitzen duen funtzioa: *adarra_da*.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



r1 = [t5, t8, t7]

r2 = [t20, t5, t9]

adarra_da(r1, *a*) = True

adarra_da(r2, *a*) = False

adarra_da([ ], *a*) = False

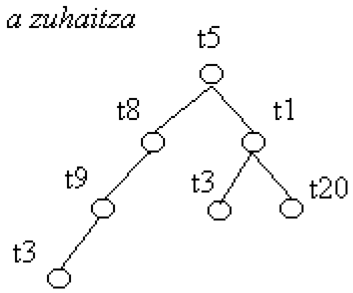
adarra_da([ ], Zhutsa) = True

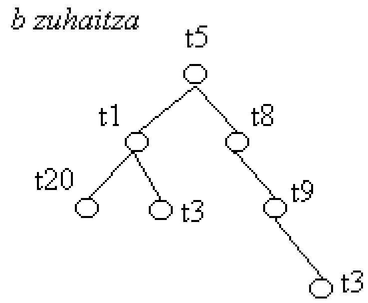
adarra_da([t4, t30, t1, t2], *a*) = False

8) izpilua_da

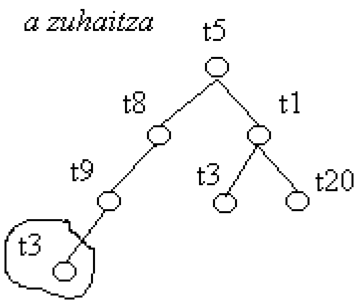
Bi zuhaitz bitar emanda, bata bestearen izpilu-irudia al den erabakitzen duen funtzioa: *izpilua_da*.

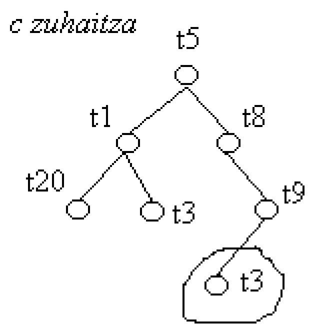
Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



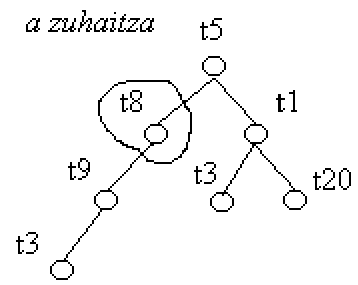


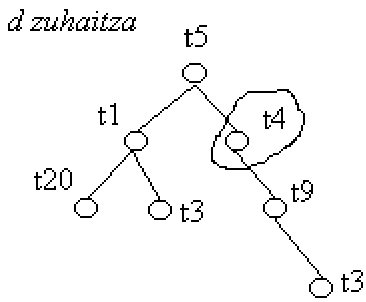
izpilua_da(*a*, *b*) = True





izpilua_da(*a*, *c*) = False





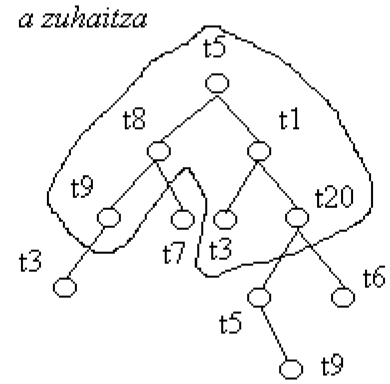
izpilua_da(*a*, *d*) = False
izpilua_da(*a*, Zhutsa) = False
izpilua_da(Zhutsa, Zhutsa) = True

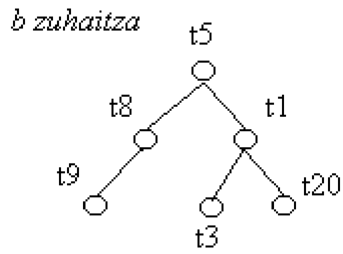
9) aurrizkia_da

Bi zuhaitz bitar emanda, lehenengoa bigarrenaren aurrizkia al den erabakitzen duen funtzioa: *aurrizkia_da*.

Hutsa ez den *v* zuhaitza eta edozein eratakoa den *w* zuhaitza hartuz, *v* zuhaitza *w* zuhaitzaren aurrizkia izateko, *w* bikoitza zuhaitzak *v* zuhaitzaren hostoetatik adabegiak erantsiz lortutako zuhaitza izan behar du. Kasu partikular bezala, *v* eta *w* berdinak badira, *v* zuhaitza *w*-ren aurrizkia izango da. Hutsa den zuhaitz bat edozein zuhaitzen aurrizkia da.

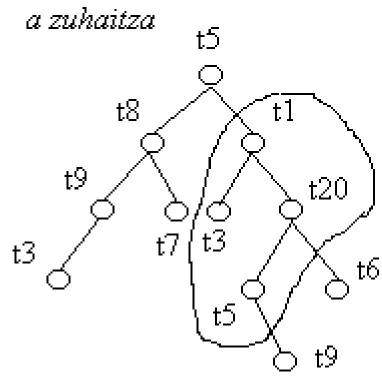
Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)

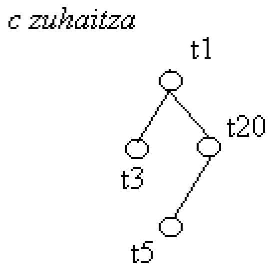




aurrizkia_da(*b*, *a*) = True

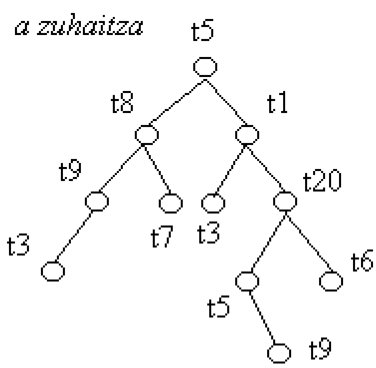
aurrizkia_da(*a*, *b*) = False

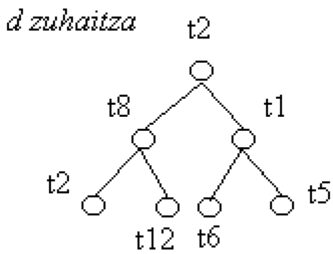




aurrizkia_da(*c*, *a*) = False

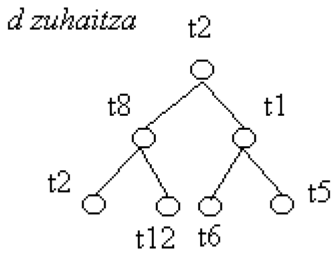
aurrizkia_da(*a*, *c*) = False

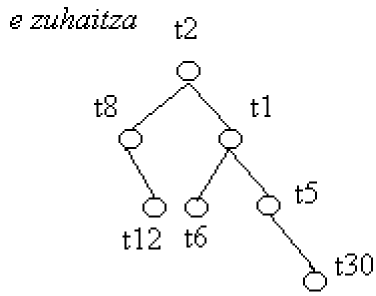




aurrizkia_da(*d*, *a*) = False

aurrizkia_da(*a*, *d*) = False





aurrizkia_da(*e*, *d*) = False aurrizkia_da(*d*, *e*) = False

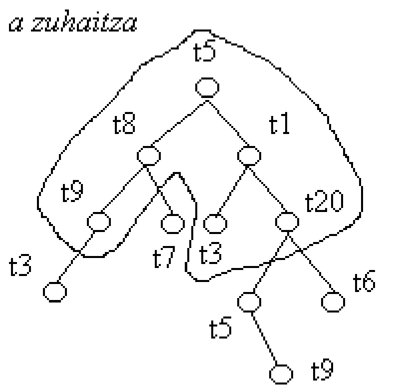
aurrizkia_da(Zhutsa, *d*) = False aurrizkia_da(*d*, Zhutsa) = False
aurrizkia_da(Zhutsa, Zhutsa) = True

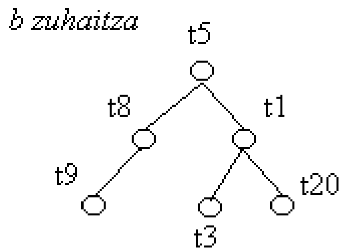
10) azpizuhaitza

Bi zuhaitz bitar emanda, lehenengoa bigarrenaren azpizuhaitza al den erabakitzen duen funtzioa: *azpizuhaitza*.

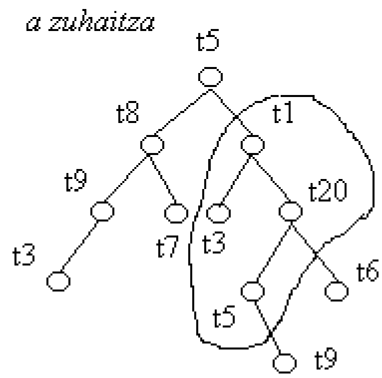
Hutsa ez den *v* zuhaitza eta edozein eratakoa den *w* zuhaitza hartuz, *v* zuhaitza *w* zuhaitzaren azpizuhaitza izateko, *w* bikoitza zuhaitzak *v* zuhaitzari adabegiak erantsiz lortutako zuhaitza izan behar du. Beraz *v* zuhaitzak *w* zuhaitzean agertu behar du bere egitura mantenduz. Kasu partikular bezala, *v* eta *w* berdinak badira, *v* zuhaitza *w*-ren azpizuhaitza izango da. Hutsa den zuhaitz bat edozein zuhaitzen azpizuhaitza da.

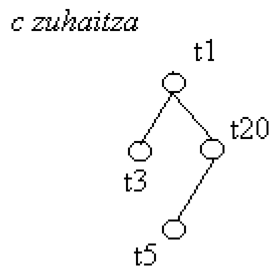
Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)





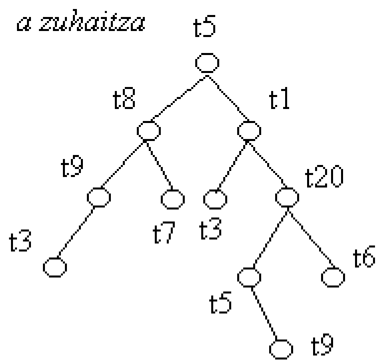
azpizuhaitza(*b*, *a*) = True azpizuhaitza(*a*, *b*) = False

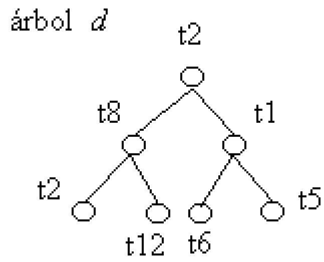




azpizuhaitza(*c*, *a*) = True

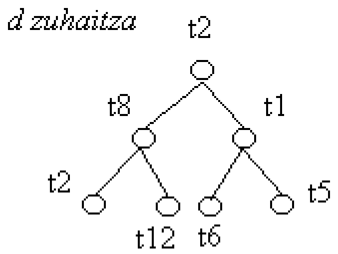
azpizuhaitza(*a*, *c*) = False

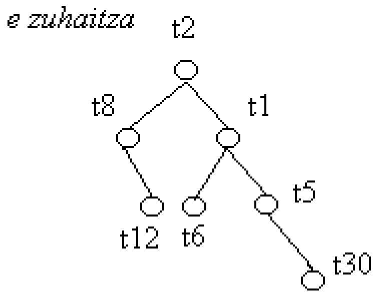




azpizuhaitza(*d*, *a*) = False

azpizuhaitza(*a*, *d*) = False





azpizuhaitza(*e*, *d*) = False

azpizuhaitza(*d*, *e*) = False

azpizuhaitza(Zhutsa, *d*) = True

azpizuhaitza(*d*, Zhutsa) = False

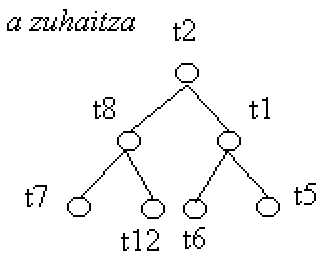
azpizuhaitza(Zhutsa, Zhutsa) = True

11) aurreord

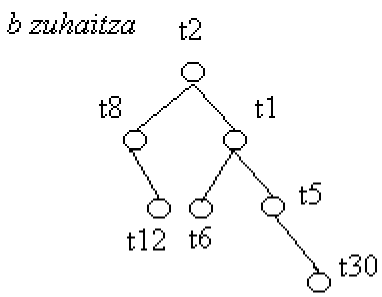
Zuhaitz bitar bat aurreordenean zeharkatuz lortzen den zerrenda itzultzen duen funtzioa: *aurreord*.

Zuhaitz bat aurreordenean zeharkatzeko hasteko erroa hartu behar da, gero ezkerreko azpizuhaitza aurreordenean zeharkatu eta bukatzeko eskuineko azpizuhaitza aurreordenean zeharkatuko da. Zuhaitza zeharkatzeko era honi aurreordena deitzen zaio lehenengo elementu bezala erroa bera hartzen delako eta azpizuhaitz bakoitzaren kasuan ere era berean jokatzen da.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



aurreord(*a*) = [t2, t8, t7, t12, t1, t6, t5]



aurreord(*b*) = [t2, t8, t12, t1, t6, t5, t30]

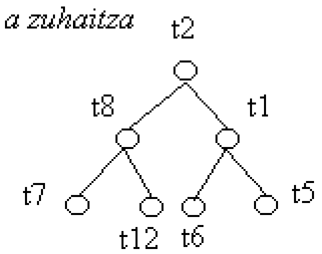
aurreord(Zhutsa) = [ ]

12) inord

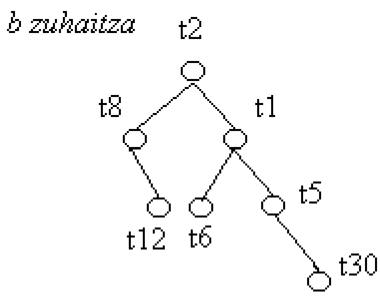
Zuhaitz bitar bat inordenean zeharkatuz lortzen den zerrenda itzultzen duen funtzioa: *inord*.

Zuhaitz bat inordenean zeharkatzeko hasteko ezkerreko azpizuhaitza inordenean zeharkatu behar da, gero erroa hartu behar da eta bukatzeko eskuineko azpizuhaitza inordenean zeharkatuko da. Zuhaitza zeharkatzeko era honi inordena deitzen zaio erroa ezkerreko azpizuhaitza zeharkatu eta gero hartzen delako kontuan eta azpizuhaitzak zeharkatzerakoan ere irizpide bera jarraitzen da.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



inord(*a*) = [t7, t8, t12, t2, t6, t1, t5]



inord(*b*) = [t8, t12, t2, t6, t1, t5, t30]

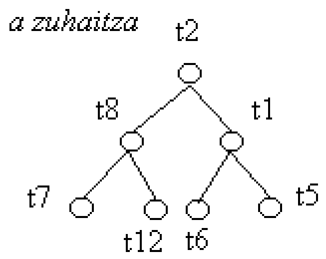
inord(Zhutsa) = [ ]

13) postord

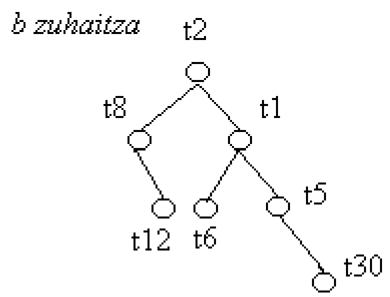
Zuhaitz bitar bat postordenean zeharkatuz lortzen den zerrenda itzultzen duen funtzioa: *postord*.

Zuhaitz bat postordenean zeharkatzeko hasteko ezkerreko azpizuhaitza postordenean zeharkatu behar da, gero eskuineko azpizuhaitza postordenean zeharkatuko da eta bukatzeko erroa hartu behar da. Zuhaitza zeharkatzeko era honi postordena deitzen zaio erroa ezkerreko azpizuhaitza eta eskuineko azpizuhaitza zeharkatu eta gero hartzen delako kontuan eta azpizuhaitzak zeharkatzerakoan ere irizpide bera jarraitzen da.

Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



postord(*a*) = [t7, t12, t8, t6, t5, t1, t2]



postord(*b*) = [t12, t8, t6, t30, t5, t1, t2]

postord(Zhutsa) = [ ]

H) Indukzio bidezko frogak zuhaitz bitarrentzat

Jarraian aipatzen diren teorema induktiboak frogatu:

1) adabegikop(a) ≤ 2^sakon(a)^ – 1

a edozein zuhaitz bitar izanda, honako hau beteko dela frogatu indukzioa erabiliz:

$$\text{adabegikop}(a) \leq 2^{\text{sakon}(a)} - 1$$

Hau da, *a* zuhaitz bitarrak gehienez $2^{\text{sakon}(a)} - 1$ adabegi ditu.

2) barnekop(a) ≤ 2^(sakon(a) – 1)^ – 1

a hutsa ez den edozein zuhaitz bitar izanda, honako hau beteko dela frogatu indukzioa erabiliz:

$$\text{barnekop }(a) \leq 2^{(\text{sakon}(a) - 1)} - 1$$

Hau da, hutsa ez den *a* zuhaitz bitarrak gehienez $2^{(\text{sakon}(a) - 1)} - 1$ barneko adabegi ditu.

3) hostokop(a) ≤ 2^(sakon(a) – 1)^

a hutsa ez den edozein zuhaitz bitar izanda, honako hau beteko dela frogatu indukzioa erabiliz:

$\text{hostokop}(a) \leq 2^{(\text{sakon}(a) - 1)}$

Hau da, hutsa ez den *a* zuhaitz bitarrak gehienez $2^{(\text{sakon}(a) - 1)}$ hosto ditu.

4) barnekop(a) ≤ hostokop(a) * (sakon(a) – 1)

a hutsa ez den edozein zuhaitz bitar izanda, honako hau beteko dela frogatu indukzioa erabiliz:

$$\text{barnekop}(a) \leq \text{hostokop}(a) * (\text{sakon}(a) - 1)$$

Hau da, hutsa ez den *a* zuhaitz bitarrak gehienez hostokop(*a*) * (sakon(*a*) – 1) barneko adabegi ditu.

5) luzera(inord(a)) = adakop(a)

a edozein zuhaitz bitar izanda, honako hau beteko dela frogatu indukzioa erabiliz:

$$\text{luzera}(\text{inord}(a)) = \text{adakop}(a)$$

Hau da, *a* zuhaitz bitar bat inordenean zeharkatuz lortzen den zerrendaren luzera *a* zuhaitzaren adabegi-kopuruaren berdina da.

6) barnekop(a) ≥ hostokop(a) – 1

a hutsa ez den edozein zuhaitz bitar izanda, honako hau beteko dela frogatu indukzioa erabiliz:

$$\text{barnekop}(a) \geq \text{hostokop}(a) - 1$$

Hau da, hutsa ez den *a* zuhaitz bitarrean barneko adabegien kopurua gutxienez hostokopurua ken 1 da.

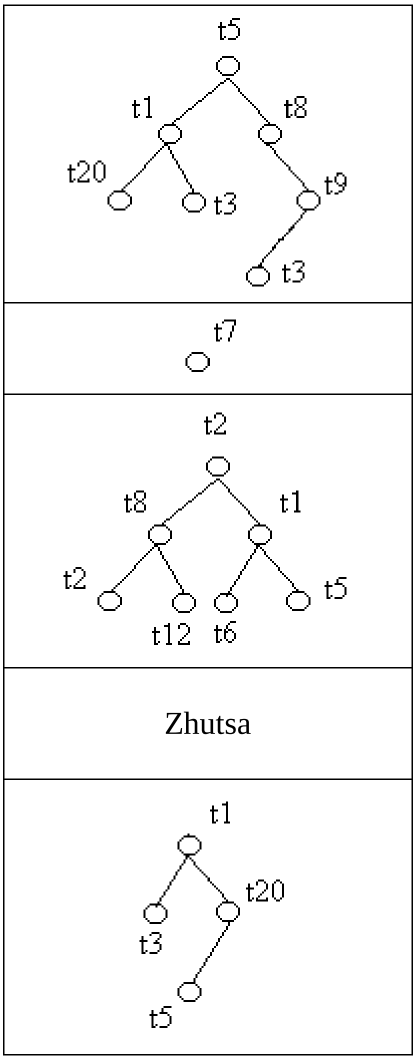
I) <u>Datu-mota desberdinak nahasian: eragiketak</u>

Jarraian aipatzen diren eragiketak definitzen dituzten ekuazioak eman:

1) zbp_aldiz

t motako zuhaitz bitarrez osatutako pila batean elementu bat zenbat aldiz agertzen den kalkulatzen duen funtzioa: *zbp_aldiz*.

1. Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



p

zbp_aldiz(t3, p) = 3

zbp_aldiz(t14, p) = 0

zbp_aldiz(t7, p) = 1

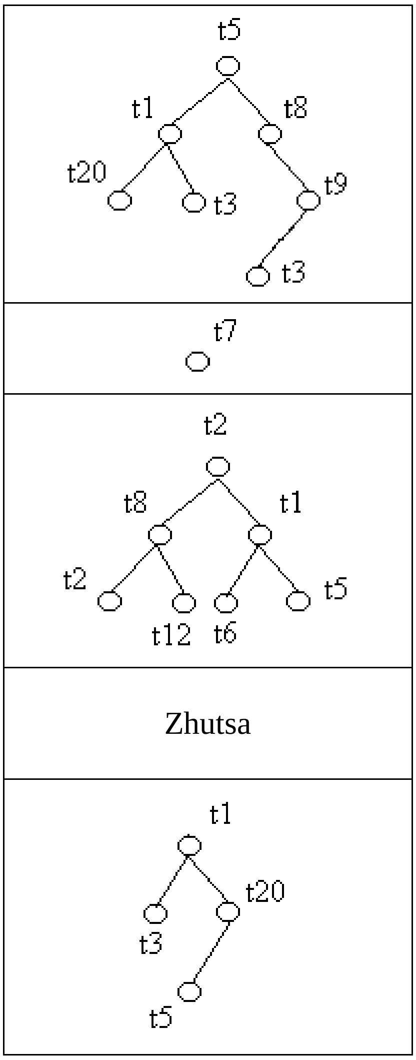
2. **Adibidea:**

zbp_aldiz(t8, Phutsa) = 0

2) zbp_zerpost

t motako zuhaitz bitarrez osatutako pila bateko zuhaitz denak postordenean zeharkatuz lortzen den zerrenda kalkulatzen duen funtzioa: *zbp_zerpost*.

1. Adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



p

zbp_zerpost(p) = [t20, t3, t1, t3, t9, t8, t5, t7, t2, t12, t8, t6, t5, t1, t2, t3, t5, t20, t1]

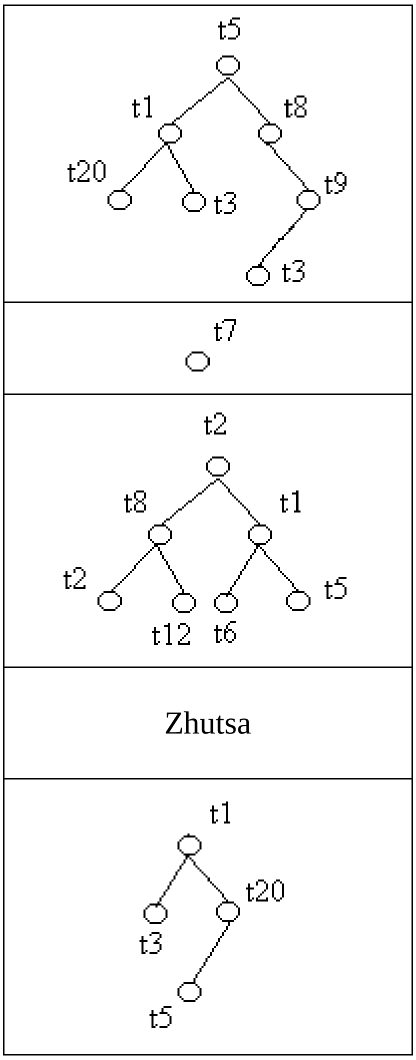
2. adibidea:

zbp_zerpost(Phutsa) = [ ]

3) zbp_hostzer

t motako zuhaitz bitarrez osatutako pila bateko zuhaitzen hostoez osatutako zerrenda kalkulatzen duen funtzioa: *zbp_hostzer*.

1. adibidea: (*ti* bezala agertzen direnak t motako elementuak dira)



p

zbp_hostzer(p) = [t20, t3, t3, t7, t2, t12, t6, t5, t3, t5]

2. adibidea:

zbp_hostzer(Phutsa) = [ ]