(Ek94) A, B eta C jokalarietako bakoitzak poker-partida bat hasi baino lehenago dolar bateko 100 billete zeukan. Baina, erne! ... A jokalaria kartazale porrokatua izateaz gain billete faltsifikatzaile amorratua dugu. Bere billete guztiak faltsuak dira!

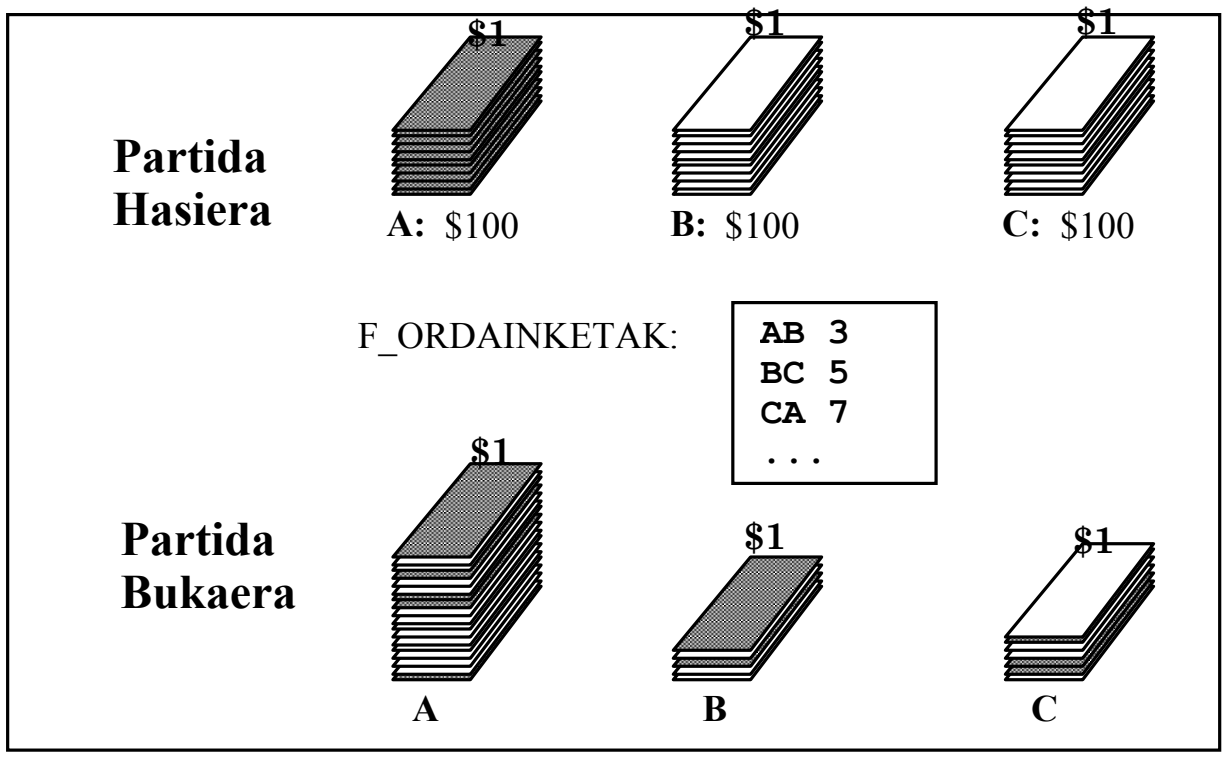
Partidan egin diren ordainketa guztiak F_PAGOS testu-fitxategian idatzi direla jakinda, egin ezazu metodo bat partida bukatutakoan A jokalari faltsifikatzaile horrek zenbat billete on eraman dituen zehaztuko duena.

F_PAGOS fitxategian lerroko ordainketa bat idatzi da, honelako formatuan: lehenengo karakterea ordaindu duen jokalariarena da (A, B edo C), bigarren karakterea kobratu duen jokalariarena eta zuriunez banatuta ondoan dator ordaindutako kopurua. Adibidez:

AB 5 A jokalariak B jokalariari 5 dolar ordaindu dio.

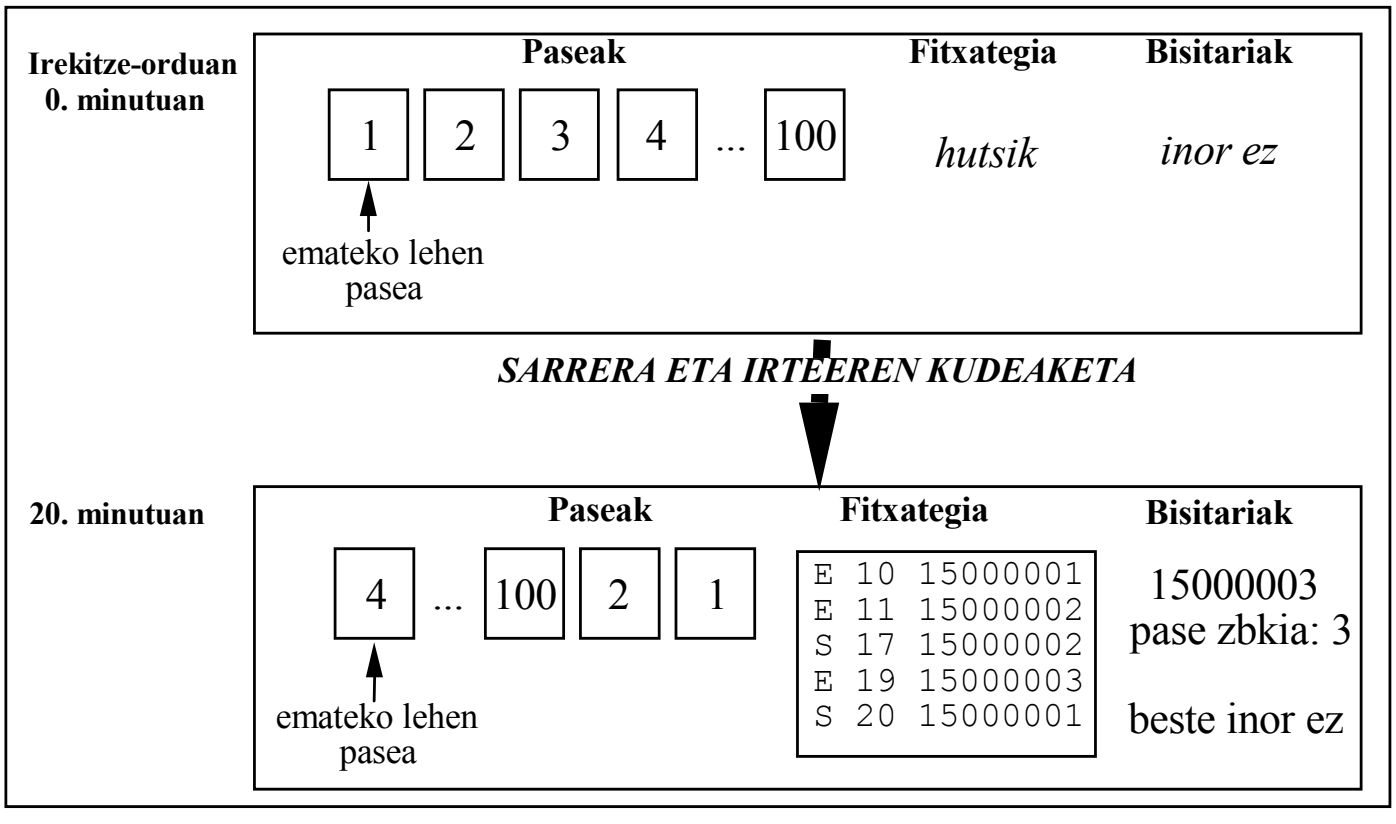
Ordainketa bat egiteko ordaintzailearen billete-multzotik gainekoa hartu eta kobratzen duenaren multzoaren gainean jartzen da behar adina aldiz.

Ezagutzen duzun datu egitura bat erabili behar duzu.



(Ir94) Zenbakidun pase bat eduki behar duzu Gugeundhen Museoan sartu ahal izateko. Harrera-gelako atezainak ematen dizu pasea eta ateratzean berari itzuli behar diozu. Atezainak fitxategi bat sortzen du pase-emate eta pase-itzultze guztiekin ondoko arauen arabera:

- Goizero zabaldu aurretik atezainak pase guztiak bata bestearen atzetik jartzen ditu goranzko ordenan 1etik 100eraino.
- Pertsona bat sartzen denean lehenengo pasea ematen dio eta sarrera-mugimendu bat sartzen du fitxategian.
  Adibidez: **`E 330 15999888`**
  mugimendu horrek esan nahi du **`15999888`** nortasun agiriko pertsona museoan sartu ('**E**',Enter) egin dela eta **`330`** minutu pasa direla museoa goizean zabaldu denetik**.**
- Pertsona bat ateratzen denean atezainak haren pasea jaso, pase-multzoko bukaeran jarri, eta irteera-mugimendua sartzen du fitxategian.
  Adibidez: **`S 360 15999888`**
  mugimendu horrek esan gura du **`15999888`** nortasun agiriko pertsona museoan irten ('**S**') egin dela eta **`360`** minutu pasa direla museoa goizean zabaldu denetik**.**



Gau batean, lapurreta-susmoak direla eta, F_MUGIMENDUAK fitxategia itxi ondoren museoko detektibeak jakin nahi du zeintzuk ziren minutu konkretu bat bukatutakoan museo barruan zeuden bisitariak eta gainera zein zen bisitari bakoitzak zeraman pasearen zenbakia (aski garrantzitsua omen da azken datu hau detektibearentzat!).

Eskatzen da:

a)Ezagutzen dituzun datu egituretakoen artean bat aukeratu pase-multzoa errepresentatzeko.

b) Honako klasea eman digute:

Bisitaria klasea

```
public class Bisitariak{
```
public void iriki ()
Bisitari-talde hutsa itzultzen du.

public void sartu(Long nan, Integer pase)
nan nortasun agiria duen pertsona *pase* zenbakiko pasearekin sartzen du bisitari-taldean.

public void irteera(Long nan)
nan nortasun agiria duen pertsona ateratzen du bisitari-taldetik

public Integer paseaIkusi(Long nan)
bisitari-taldean dagoen eta *nan* zenbakiko nortasun-agiria duen bisitariaren pasea itzultzen du

public boolean norbaitDago()
False itzultzen du bisitari-taldean inor ez badago eta true bestela

Bisitariak klasea eta pase-multzorako aukeratu duzun datu egitura erabiliz parametriza, diseina eta inplementa ezazu honelako prozedura bat: minutu bat eta F_MUGIMENDUAK fitxategia emanda irteera estandarrean idatziko ditu minutu hori bukatutakoan museo barruan zeuden bisitarien nortasun-agiria eta pasearen zenbakia.

(Ek95) *MUTSOBITSI COMPUTERS*-eko kalkuku-zentroan inprimatze-lanak **lps1**, **lps2**, **lps3**, **lps4** eta **lps5** inprimagailuetan egiten dira arazorik ez dagoenean.

Elektrizitatea pikutara joaten denean inprimagailu horiek guztiak deskonektatuta geratzen dira, baina segurtasun-sistemak **lps6** larrialdietarako inprimagailua piztu eta bertara bidaltzen ditu ohizko inprimagailuetan zintzilik geratu diren inprimatze-lan guztiak.

Itzalketa-kasuan honela birrantolatuko dira inprimatze-lanak: **lps1**-etik hasita eta **lps5**-ean bukatuz inprimagailuetako lan guztiak zeuden ordena berean eramango dira **lps6** inprimagailura.

Itzalketa konpontzen ez den artean **lps1**, **lps2**, **lps3**, **lps4** eta **lps5** inprimagailuetara etorriko diren inprimatze-lanak **lps6** larrialdi-inprimagailura eraman beharko dira.

Itzalketa konpondutakoan ohizko inprimagailuak berriro pizten dira lan berriak hartzeko moduan. Hala ere, **lps6**-ra eramandako lan guztiak bertan geratuko dira inprimatuak izan arte.

Eskatzen da:

a)**Ezagutzen dituzun datu egituretakoen artean bat aukeratu** inprimagailu baten inprimatze-lanak errepresentatzeko.

b) *Aukeratutako* DMAa erabiliz, **diseina eta inplementa ezazu honelako programa bat: F_INP.dat** fitxategiko datuak kontuan hartuz, terminalean erakutsiko ditu azkeneko egoeran inprimagailu bakoitzean inprimatzeko geratzen diren lanak .

Ematen diguten **F_INP.dat** fixategian inprimatze-gertaera guztiak azaltzen dira. Lau gertaera mota daude: Eskaerak (E), Bukaerak (B), Itzalketak (I) eta Konponketak (K).
Adibidea:

F_INP.dat

```
(1)→ Elps2 1
     Elps3 2
     Elps5 3
     Elps5 4
(2)→ Blps5
     Elps2 5
(3)→ I
     Blps6
     Elps1 6
(4)→ K
     Elps1 7
```

F_INP.dat fitxategiko **(1)** gertaerak inprimatzeko eskaeraren (E) berri ematen du. Eskatzen da 1 zenbakiko lana lps2 inprimagailuan inprimatzea

(2) gertaerak **lps5** inprimagailuan inprimatzen ari den lana bukatu (B) egin dela adierazten du

(3) gertaerak itzalketa (I) baten hasiera adierazten du

(4) –(k) itzalketa konpondu (K) egin del adierazten du

Lehengo adibidean, **(2)** gertaera eta gero, honakoa da inprimagailuen egoera:

lps1	**lps2**	**lps3**	**lps4**	**lps5**	**lps6**
-	1	2	-	4	-

Itzalketa, **(3)**, gertatu baino lehen honako egoera hau zegoen:

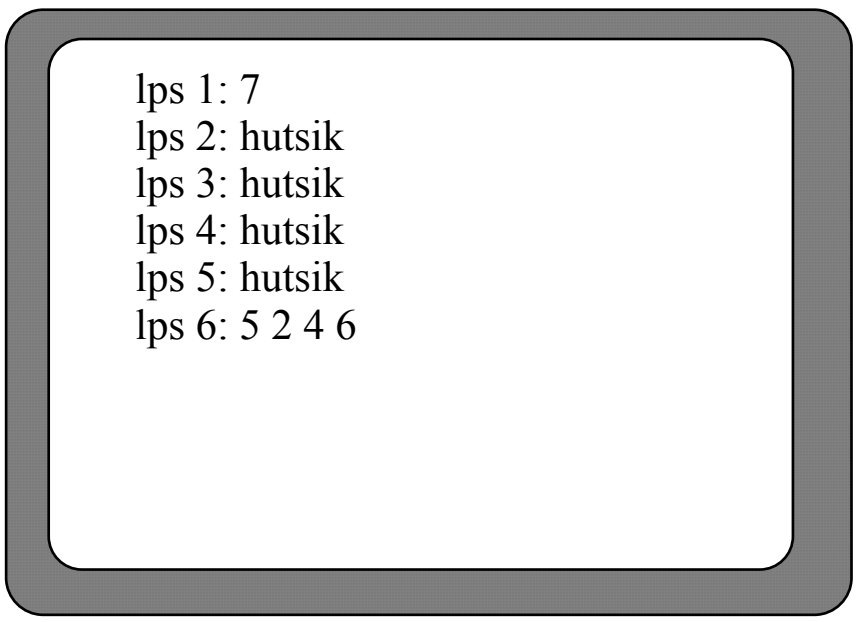
lps1	**lps2**	**lps3**	**lps4**	**lps5**	**lps6**
-	1	2	-	4	-
	5				

Itzalketaren ondorioz egindako berrantolaketa (gorago definitu dena) bukatutakoan, hau da, **(3)** gertaera prozesatu ondoren, honela geratzen dira lanak:

lps1	**lps2**	**lps3**	**lps4**	**lps5**	**lps6**	
-	-	-	-	-	1	(lps2-koa)
					5	(")
					2	(lps3-koa)
					4	(lps5-koa)

F_INP.dat fitxategiko datuak egokiak direla suposatu. Hau da, ez da azalduko bigarren itzalketa bat tartean konponketarik egon ez bada, itzalaldian ez da lan-bukaerarik sortzen ohizko inprimagailuetan, ...

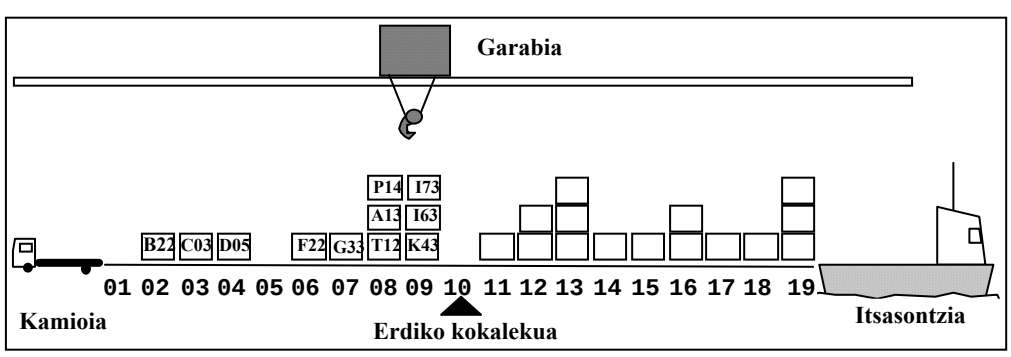
Eskatutako programak aurreko adibiderako pantailan idatzi beharko duena ondokoa da:



1. **(Ir95)** Pasaiako portuan garabi handi bat dago kontenedoreak mugitzeko. Garabiak kontenedoreak *kamioietatik* deskargatzen ditu, kaian utzi eta behar denean *itsasuntzian* kargatzen ditu .

   Kaian 19 kokaleku desberdin markatu dira kontenedoreak uzteko. Garabiaren lana kontenedoreak mugitzea da edozein kokalekutik beste edozeinetara, baita kamioitik kokaleku batera edo kokalekuetatik itsasontzira ere.

   Erdiko kokalekua berezia da. Mugitu behar den kontenedore bat gainean beste batzuk dauzkadanean, erdiko kokalekua hau erabiltzen da, une batez gainean zeuden kontenedoreak alboratzeko. Kontenedorea mugitu ondoren kokaleku lagungarri horretan utzi diren kontenedoreak jatorrizko kokalekura itzultzen dira.



Eskatzen da:

a) **Ezagutzen dituzun datu egituretakoen artean bat aukeratu** kokaleku batean bildutako kontenedore-multzoa errepresentatzeko. Azaldu zergatik.
b) *Aukeratutako* datu egitura erabiliz, **diseina eta inplementa ezazu** F_GARABIA.DAT testu-fitxategiko karga- eta deskarga-aginduen ondorioz garabiak egin dituen kontenedore-mugimendu guztiak bilduko dituen F_GARABIA.IRT fitxategia sortuko duen **programa bat**. Kaiaren hasierako egoera F_HASIERAKOAK.DAT fitxategiaren bitartez lortzen da.

Suposa daiteke F_GARABIA.DAT fitxategiko karga- eta deskarga-agindu guztiak onargarriak direla, ez dela sortuko errorerik beraiek exekutatzean.

<u>Adibidea</u>:

Irudiko egoera lortzeko honako hasierako fitxategi hau erabili da:

F_HASIERAKOAK.DAT

(1) →

```
B22 2
C03 3
D05 4
F22 6
G33 7
T12 8
A13 8
P14 8
K43 9
I63 9
I73 9
...
```

F_HASIERAKOAK.DAT fitxategiko **(1)** lerroak hasierako egoeran `B22` kontenedorea **2.** kokalekuan dagoela adierazten du.

Kamioietatik deskargatu ziren ordena berean agertzen dira kokaleku bakoitzeko kontenedoreak. Beraz, kokaleku jakin baterako lehenengo aginduko kontenedorea lurra ikutzen dagoena da, eta kokaleku jakin horretarako azken aginduko kontenedorea goiko muturrean dagoena da.

Karga eta deskargen fitxategia ondokoa baldin bada:

F_GARABIA.DAT

```
(1)→ D B02 3
     D T12 1
     D B04 1
     D P00 1
     D P02 18
(2)→ K T12 8
     D Z07 3
     K I63 9
     ...
```

F_GARABIA.DAT fitxtegiko **(1)** lerroak **B02** kontenedorea **3.** kokalekuan **deskarga** tzeko-agindua (**D**) adierazten du . Deskargetan kamioitik hartzen da kontenedorea.

(2) lerroak **8.** kokalekuan dagoen **T12** kontenedorea itsasontzian **karga** tzeko (**K**) agindua adierazten du

lortu beharko den fitxategia honako hau da:

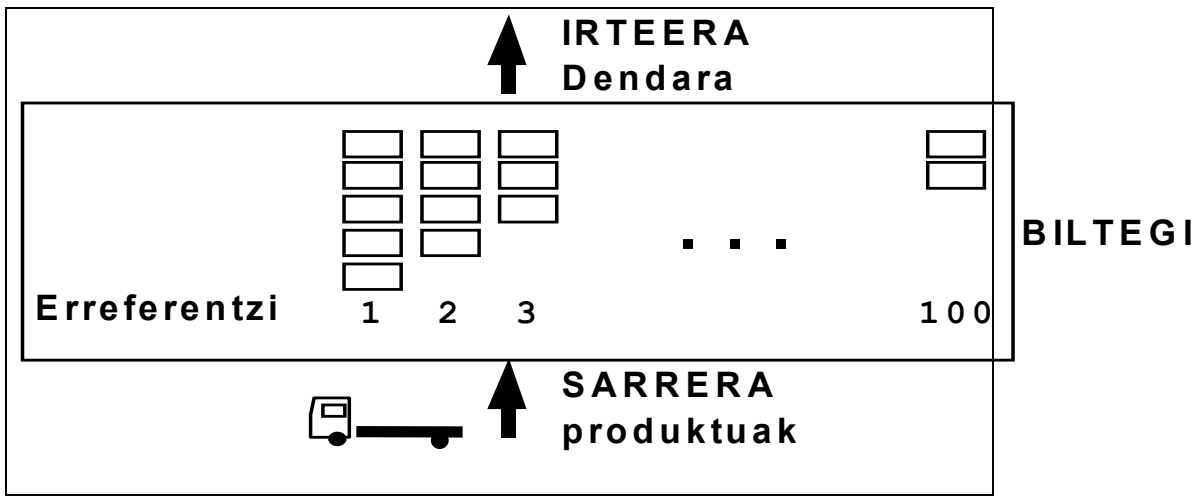
F_GARABIA.IRT

```
Eraman kontenedore bat Kamioitik        3.era
Eraman kontenedore bat Kamioitik        1.era
Eraman kontenedore bat Kamioitik        1.era
Eraman kontenedore bat Kamioitik       11.era
Eraman kontenedore bat Kamioitik       18.era
Eraman kontenedore bat    8.etik       10.era
Eraman kontenedore bat    8.etik       10.era
Eraman kontenedore bat    8.etik Itsasontzira
Eraman kontenedore bat   10.etik        8.era
Eraman kontenedore bat   10.etik        8.era
Eraman kontenedore bat Kamioitik        3.era
Eraman kontenedore bat    9.etik       10.era
Eraman kontenedore bat    9.etik Itsasontzira
Eraman kontenedore bat   10.etik        9.era
...
```

(Ek96) SALDOSKI hipermermakuak bere produktuen automatizazioa egin nahi du. Hasieran, eta frogra bezala 100 produktu hartuko ditu.

Produktu guztien kaxa sarrera eta irteerak **F_MUGIMENDUAK.DAT** fitxategian gordetzen dira automatikoki. Biltegitik sartu edo irtetzen den artikulo kaxa bakoitzengatik bere erreferentzia eta iraungipen data idazten da.

Biltegiaren produktuen kaxa irteera eta sarreraren irizpidea hurrengoa da. Biltegitik aterako dira, iraungipen data gertuen duten produktuek. Ez da onartuko sarrera bezala inongo produkturik, bere iraungipen data biltegian dagoen produktuaren iraungipen data baino gerokoa bada.



Se pide:

a) **Ezagutzen dituzun datu egituretakoen artean bat aukeratu** biltegia errepresentatzeko.

b) Aukeratutako klasea erabiliz, **diseina eta inplementa** ezazu metodo bat. Metodoak bi fitxategi edukiko ditu sarreran:

- **F_HSIERAKOA.DAT fitxategia, hasierako egoera gordetzen duen datuekin.**
- **F_MUGIMENDUAK.DAT fitxategia, biltegian egon diren irteera/sarrerazko transakzioekin.**

Eta irteeran inprimatuko du:

- **<u>Hiru egunean baino gutxiago</u> iraungitzen diren produktuen zerrenda**
- **Produktu berrien eskaerak, beren unitateak 5 baino gutxiago badira**

Suposatzen da **F_MUGIMENDUAK.DAT** dauden eragiketa guztiak zuzenak direla.

(Ir96) Villa Garcia-ko posta bulegoa, 10 postari dauzka gutun guztien banaketa egiteko. Postari bakoitzak, barruti bateko gutun guztiak banatzen ditu (**10.001tik, 10.010ra**) soilik.

Postarien artean, gutun banaketa horrela egiten da:

- Gutunen banaketa, ailegatze ordenean egiten da.

- Postari bakoitzak gehienez 200 gutun banatu dezake.

- Egunero, gutunen banaketa egiten da postarien artean, barrutiaren arabera. Hau da, gutunaren barrutia 10004 bada, postari horretan dagoen postariari esleitzen zaio. Metodo hau jarraitzen da, postariak dauzkan gutunak 200 baino gutxiago badira.

- Metodo hau jarraituz banatu ezin daitezkeen gutunak, postarien artean banatuko dira beste irizpide desberdin bat erabiliz.

Correos-eko bulegokideek programa bat egin nahi dute, aurkeztutako banatze prozesua aurrera eramateko. Programaren helburua, postari bakoitzaren gutun zerrenda eta banatu gabe geratzen diren gutunen zerrenda lortzea da.

Eskatzen da:

a) Aukeratu datu egitura egokiena gutunen banaketa aurrera eramateko eta jarraian eskatzen dizkiguten txostenak inprimatzeko. Kontuan hartu, erabili daitekeela ezagutzen diren datu egiturak.
b) Aukeratutako egitura erabiliz, diseina eta inplementatu metodo bat. Metodo honek **gutunak ailegatu diren ordenean** duen informazioa **KARTAK.TXT** fitxategia irakurriko du, eta gutunen hurrengo infomazio inprimatuko du, **barrutiaz taldekatuta** eta barruti baten barnean, **sarrerazko ordena mantenduz**:
• Alde batetik, postari bakoitzari esleitutako gutunak..

• Beste aldetik, postaririk gabe geratu diren gutunak.
Barruti bakoitzarentzat, (gutun guztiak inprimatu baino lehen), postariak falta zaizkion gutunen kopurua 200 arte ailegatzeko inprimatuko da. 200 gutunera ailegatzen bada, barruti horretan banatu gabe geratzen diren gutunen kopurua inprimatuko da

KATAK.TXT fitxategiaren egitura hurrengoa da: (gutunak ailegatzeko ordenean) :

```
10002 Gran vía 3
10002 Bailén 20
10001 Sicilia 11
10002 Gran vía 3
  ...
```

Banaketaren irteera (helbideak barrutiaz taldekatuta. Taldearen barruan, sarrerako ordenean)

postari 10001
10001 Sicilia 11
10001 ...

postari 10002
10002 Gran vía 3
10002 Bailén 20
10002 ...

postari 10003
10003 ...

Geratzen diren gutunen irteera (helbideak barrutiaz ordenatuta. Taldearen barruen, sarrerako ordena)

Barrutia 10001: 2
10001 Ortega 7
10001 Velazquez 4

Barrutia 10002: -5

Barrutia 10003: 0

Barrutia 10004: 4
10004 Gaudi 3
10004 Urdaneta 20
10004 Ramon y Cajal 30
10004 Cervantes 7

Barrutia 10005: -10
...

→ 2 gutun geratzen dira banatzeko, Postariak 200 gutun dauzka

→ Gutun guztiak banatuta eta Postariak 195 dauzka

→ Gutun guztiak banatuta eta Postariak 195 dauzka

(Ek97) Eskoziako gaztelu zahar batean mamu famili bat bizi da (mamu bakoitza NAN zenbaki bat dauka). Azkeanaldi hauetan, gazteluko nagusieek **25** logelazko hotel bat egitea erabaki dute.Izutuak emateko helburuarekin, mamuak logelaz antolatu dira ondoko eran:

- Mamu bakoitzak logela bat dauka bere mempean.
- Logela bakoitzak, mamu bat baino gehiago eduki dezake.
- Logela bakoitzean izutu bakar bat ematen da egunero (mamuren bat badago, beste kasuan, ez da izuturik emango).

Mamuek orden zorrotz batean lan egiten dute: Logela bat alokatuta dagoenean, logela horretako lehen mamuak izutua emanten du, eta ez du beste izuturik emango, logela horretan dauden beste mamu guztiek izutu bat eman arte.

Mamu guztiek izutu kopuru berdina daukate: **10**. Mamu batek 10 izutu eman ondoren, bizitza hobeagora joaten da eta gaztelua uzten du.

Gazteluko bulegoan, **HASIERA.TXT** textu fitxategian, mamuen banaketa logeletan, eta mamu bakoitzak eman dituen izutu kopurua gordetzen da. Lehen agertzen diren mamuak, izutuak lehen eman ditzaketenak dira. **ERRESERBAK.TXT**, erreserba guztien informazioa agertzen da.

Eskatzen da:

a) Aukeratu **datu egitura** egokiena(k) irteera lortzeko. Datu egitura bakoitzentzako, elementuen datu mota adierazi..

b) Aukeratutako datu egiturak erabiliz, metodo bat garatu. Metodo honek, **HASIERA.TXT** eta **ERRESERBAK.TXT** fitxategiak irakurriko ditu, eta hurrengo datuak argitaratuko ditu pantailatik:

- Bizitza hobeagora pasa diren mamuen NAN-a.
- Bizitza hobeagora mamu gehiago pasa diren logela zenbakia. (Bat baino gehiago egongo balitz, edozein).

INICIO.TXT fitxategiak, lerro bat dauka mamu bakoitzarentzat, izutuak **emanteko ordenean sailkatua**. Lerro bakoitzak, mamuaren NAN-a, orain arte emandako izutu kopurua eta bere logela gordetzen du.

```
12554231 2    14
13455904 1    25
15667231 8    1
...
19563211 9    14
```

mamuaren-Nan emandako izutuak logela zenbakia

ERRESERBAK.TXT fitxategiak, lerro bat dauka erreserba bakoitzengatik, **ostalariak ailegatzeko ordenean**. Lerro bakoitzak, logela zenbakia, eta egon behar diren egun kopurua gordetzen du

```
25    3
12    5
10    1
...
5     4
```

logela zenbakia erreserbatutako gauak

Pantailatik, bizitza hobeagora pasa diren mamuak inprimatuko dira (10 izutu) bizitza hobeagora mamu gehiago pasa diren logela zenbakia

```
Bizitza hobeagora pasa diren mamuak:
13455904
19563211
...
bizitza hobeagora mamu gehiago pasa diren logela zenbakia:
14
```

(Ir97) Fakultatean telefonozko sistema berri bat instalatu dute. Sistema honek, deialdiaren igorlearen zenbakia, deitutako luzapenaren zenbakia, eta elkarrizketaren denbora gordetzen du.

Eskatzen digute, fakultatetik kanpo, **denbora gehiago hitzegiten den telefonoaren** (elkarrizketa guztien denbora batura) zenbakia. Baita ere, **telefono horretara egin diren deialdi guztien informazioa**, hau da, luzapenaren zenbakia eta elkarrizketaren denbora, *egin ziren ordenean*.

Eskatzen da:

a) Aukeratu datu egitura egokiena sarrera lortzeko, suposatuz ***gehienez 100 telefono*** egongo direla eta bere zenbakiak jarraian egongo direla. (adibidez 943000000tik 943000100ra).
Ezagutzen duzun datu egitura bat erabili, eta elementuen datu mota adierazi.

b) a) atalean aukeratutako egiturekin, metodo bat garatu, datuak **DEIALDIAK.TXT fitxategitik irakurtzeko eta eskatutako emaitzak inprimatzen duena.**

c) Zer egitura aukeratuko zenuke elementu kopurua gehienez 100 izango EZ balitz. Pentsatu ezagutzen duzun datu egitura ezagun batean.

DEIALDIAK.TXT fitxategiak deialdi guztien informazioa gordetzen du. Deitzen den zenbakiari (9 digitu), luzapena(3 digitu) eta denbora.

```
948750000 314 2
943218000 235 10
948750000 314 3
943007789 135 3
948750000 093 5
944700988 314 12
943218000 211 7
948750000 122 30
...
```

Irteera:

```
zenbakia: 948750000
denbora: 2000

Deialdi zerrenda:
314 2
314 3
093 5
122 30
...
```

(Ir98) ZerrendaOrdenatuak klasearen espezifikazioa ematen digute. Elementuak osoak dira eta bere inplementazioa 2 pila erabiliz egin da.

```
public class zerrendaOrdenatuak {
private Stack pila1=new Stack();
private Stack pila2=new Stack();

-- Zerrendak hasieratzen ditu
public void hasieratu()
-- elem elementua zerrendan ordenatua txertazen du
-- elem zerrendan badago, berriz txertatzen du
public void txertatuOrdenatua(int pValor)
-- elem elementuaren agerpen guztiak ezabatzen ditu
-- elem agertzen ez bada, zerrenda berdin geratzen da
public void ezabatuOrdenatua(int pValor)
-- Zerrendaren lehen elementua itzultzen du
public int getLehena ()
-- Hasieran bezalako zerrenda bat itzultzen du, baino lehenengo elementurik gabe.
public void lehenaEzabatu ()
-- true zerrenda hutsa bada, false beste kasuan
public boolean isHutsa()
-- Zerrendaren edukia inprimatzen du
public void print ()
```

Eskatzen da:

a) Irudi bat egin, pila1 eta pila2-ren balioekin, zerrendak 1,2,3,3,4 balioak duenean.
b) Gertatu ahal daiteke, zerrendak 1,2,3,3,4 elementuak edukitzea eta a) atalean ipini dituzun elementuak beste orden batean egotea? Zergatik?
c) isHutsa eta ezabatuOrdenatua metodoen **diseina eta inplementatu.**
d) Inplementazio hau, ezagutzen dugun zerrenda estekatu baten inplementazioarekin konparatzen dugu. Konparatu eragiketa guztien konplexutasun ordena bi inplementazioetan. (Esanez zein den kasu bakoitzean)
e) Bietatik, zein errepresentazio aukeratuko zenuke?. Zergatik?

EBAZPENAK

Ariketa guztien ebazpenerako **FitxategiIrakurlea** klasea ematen da fitxategien irakurketa laguntzeko. Klase honen interfaze publikoa hurrengoa da.

```
public class FitxategiIrakurlea {

 //Metodo eraikitzailea, parametro String bat fitxategiaren izenarekin
  public FitxategiIrakurlea (String file) {}

  // True itzultzen du fitxategian lerro gehiago badaude eta false beste kasuan
  public boolean hasMoreTokens() {}

  //Data objektu bat itzultzen du lerro baten informazioarekin. Objektu honen
  //edukia aplikazioaren kontextuan definituko da.
    public Data getToken() {}
}
```

Adibide bezala, ek94 ariketan, *Data* klasearen espezifikazioa hurrengoa izango litzateke:

```
public class Data {

private int jokalari1;
private int jokalari2;
private int kantitatea;

  //Eraikitzailea
  public Data(int pJok1,int pJok2,int pK)

  //Lehendabiziko jokalaria itzultzen du
  public int getJok1() {}

  //Bigarrengo jokalaria itzultzen du
  public int getJok2() {}

  //Emandako bilete kopurua itzultzen du
  public int getKant() {}
  }
```

(Ek94)

Espezifikazioa:
--Pre: F_ORDAINKETAK fitxategian ordainketa lerroak daude. Adibidez:“AB 5”,
-- A jokarariak B jokalariari 5$ ordaindu dizkiola esan nahi du.
-- Ordainketak banan banan egiten dira, sortaren gaineko biletea hartuz.
-- Fitxategian dauden ordainketa guztiak egin daitezke (jokalariak bileteak dauzka
-- eskatzen den eragiketa egiteko).
--Pos: Pantailan, A jokalariak eraman dituen bilete legalak inprimatzen dira.

Datu egitura(k):
Pila motako 3 elementuko Array-a, jokalariengatik indexatuta, bilete sortak errepresentatzeko. Pilak bileteak dauzkate. Bilete bakoitzak bi aldagai dauzka. Bere kantitatea eta bere benetakotasuna.

Algoritmoa:

1. Bilete sortak hasieratu
2 Fitxategiko ordainketak kudeatu
3.Zenbatu faltsifikatzaileen bileteak

Bilete sortak hasieratu
Jokalari guztientzat
    Bere pila hasieratu
    Bere 100 bilete empilatu
    (faltsuak bakarrik A jokalarienak).

2. <u>Fitxategiko ordainketak kudeatu</u>
Fitxategia ireki
While (datuak fitxategian)
    Datuak irakurri
    2.1. Ordainketa bat kudetatu
endWhile
fitxategia itxi

2.1. <u>Ordainketa bat kudeatu</u>
Ordaintzeko bilete bakoitzentzat
    Ordaintzen duenengatik ezabatu
    Kobratzen duenari txertatu

<u>3. Zenbatu faltsifikatzaileen bileteak</u>
While (billeteak A sortan)
    If (goiko biletea egiazkoa da)
      zenbatu
    Biletea ezabatu
endWhile

Inplementazioa:

Erabilitako osagarri klaseak

```
public class Billete {

  private int q;
  private boolean b;

  public boolean originala()
}
```

Programa Nagusia

```
public static void main(String[] args) {
    Game g=new Game("c:\\DataStructures2\\data\\j94\\j94.txt");
    g.partidaJokatu ();
    g.datuakInprimatu();
  }


public class Game {
FitxategiIrakurlea file;
Stack<Billete>[] dirua= new Stack<Billete>[4];

  public Game(String f) {
   for (int i=1;i<4;i++)
    dirua[i]=new Stack<Billete>();
   pilakHasieratu();
   file=new FitxategiIrakurlea(f);
  }
  public  void pilakHasieratu() {
   boolean originala;
   for(int j=1;j<=3;j++) {
    if (j==1) originala=false;
     else originala=true;
    for (int b=1;b<=100;b++)
      dirua[j].push(new Billete(1, originala));
    }
  }
  public void partidaJokatu() {
   Data d;
   int j11,j12,qq;

   while (file.hasMoreTokens()) {
    d=file.getTokens();
    j11=d.getJok1();
    j12=d.getJok2();
    qq=d.getKant();
    for (int nb=1; nb<=qq;nb++)
     dirua [j12].push(dirua[j11].pop());
   }
   }
  public void datuakInprimatu() {
   Billete b;
   int q=0;
Enumeration<Billete> e= dirua[1].elements();
while (e.hasMoreElements()) {
 b=e.nextElement();
```

```
    if (!b.originala()) q++;
   }
   System.out.println("Jokalari   faltsifikantea   "+q+"   billete
orignalak lapurtu ditu");
  }}
```

(Ir94)
Espezifikazioa:

```
public static void susmagarriak(int min)
```

Pantailan, museoan "min" minutuan zeuden pertsonen NAN zenbakia bere sarrera baimen zenbakiarekin inprimatuko dira.

Datu egitura(k):
Ilara bat erabiltzen dugu sarrera baimen zenbakiak gordetzeko (osoko ilara). Hasieran, 1etik 100era ordenatuta egongo dira.

1. Sarreren ilara hasieratu
2. Fitxategiaren mugimenduak kudeatu emandako minutu arte
3. Museoan dauden bisitariak idatzi

1. Sarreren ilara hasieratu
Bisitariak iriki
Baimenen ilara hasieratu
Txertatu 100 baimen ilaran

2. Fitxategiaren mugimenduak kudeatu emandako minutu arte
Fitxategia iriki
While (mugimenduak eta minutua ez ailegatu)
    Mugimendua irakurri
    2.1.Mugimendua kudeatu
endWhile
fitxategia itxi

2.1. <u>Mugimendua kudeatu</u>
If (irakurritako minutua <emandako min)
    If (sarrera)
        Ilararen baimena hartu
        Bisitaria txertatu
    else *(irteera)*
        Bisitaria atera
        Baimena ilaran gorde

3. <u>Museoan dauden bisitariak idatzi</u>
While (elementuak)
    NAN eta baimena hartu
    NAN eta baimena idatzi
    Hurrengo bisitarira joan

Inplementazioa:

```
 public static void main(String[] args) {
   String dataFile="c:\\DataStructures2\\data\\s94\\s94.txt";
   Museum m=new Museum(dataFile);
   m.susmagarriak(40);
   m.printPersonsInMuseum();
  }

public class Museum {
 Queue<Integer> paseak = new ArrayQueue<Integer>();
 Bisitariak bisit=new Bisitariak();
 FitxategiIrakurlea file;
  //Eraikitzeilea
  public Museum(String f) {
    file=new FitxategiIrakurlea(f);
  // ilara hasieratu
    bisit.iriki();
    for (int p=1;p<=100;p++){
     paseak.enqueue(new Integer(p));
     }
  }
  public  void susmagarriak(int minutu) {

     Data d=new Data('N',-1,new Long(0));
    while ( (d.getTs()<minutu) && (file.hasMoreTokens()) ){
    d=file.getTokens();
     if (d.getTs()<=minutu)
      if (d.getOp()=='E')
       {
          System.out.println("Sarrera"+d.getOp()+""+d.getTs()+"
                                                    "+d.getId());
        bisit.sartu(d.getId(),paseak.dequeue());
       }
         else
           {
           System.out.println("Irteera"+d.getOp()+"
                                  "+d.getTs()+" "+d.getId());
           paseak.enqueue(bisit.paseaIkusi(d.getId()));
           bisit.atera(d.getId());
           }
  }
}
public void printPersonsInMuseum() {
     //Idatzi museoan gertatzen diren pertsonak
     System.out.println("Geratzen dira");
     Enumeration<Long> en=bisit.getKeys();
     while (en.hasMoreElements()) {
     Long nan=en.nextElement();
     System.out.println(nan+" "+visit.hs.get(dni));
     }
}
 }
```

(Ek95)

Espezifikazioa:
--Aurre: F_INP.dat fitxategian, sistemak lanegun batean zehar eduki dituen inpresiozko eskaera guztiak kronologikoki gordeta daude. “Plps2 1”, lps2 inpresoran 1 fitxategiaren inpresio eskaera errepresentazen du. “Flps2”, lps2 inpresoraren eskaeraren bukaera errepresentatzen du. “A” (Apagon) itzalaldi esan nahi du eta “S” itzalketa konponduta.
--Pos: Pantailan, sistemaren inpresora bakoitzak inprimatu gabe dauzkan eskaeraren egoera agertuko du

Datu egitura(k):
6 ilarako array bat inpresora izenengatik indexatuta. Ilararen elementuak osoak dira(eskaeraren zenbakia).

Algoritmoa:

```
1.Ilarak hasieratu
2 Fitxategiaren eragiketak kudeatu
3.Ilara guztien egoera inprimatu
```

```
2. Fitxategiaren eragiketak kudeatu
 Fitxategia iriki
Sistemaren egoera ez itzalaldi ipini
While (elementuak fitxategian)
        Eragiketa irakurri
        2.1.Eragiketa kudeatu
endWhile
fitxategia itxi
```

```
2.1Eragiketa kudetatu
If eragiketa E
        Impresora eta eskaera irakurri
        2.1.1. eskaera impresoran ipini
else if eragiketa B
        inpresora irakurri
        inpresoraren lehen eskaera ezabatu
else if eragiketa I
        itzalketa egoeran ipini
        2.1.2.eskaera guztiak larrialdiko inpresoran ipini
else (eragiketa K)
        ez itzalaldi egoeran ipini
end
fitxategiren hurrengo lerrora joan
```

```
2.1.1. eskaera inpresoran ipini
if itzalaldi egoera
        larrialdiko inpresoran ipini
else
        irakurritako inpresoran ipini
```

```
2.1.2. eskaera guztiak larrialdiko inpresoran ipini:
for (ilara guztientzat (larrialdi izan ezik))
        ilara hustu larrialdiko inpresoran
```

```
3. Ilara guztien egoera inprimatu
Ilara guztientzat
        Bere izena inprimatu
        If (eskaerak ilaran)
                Edukia inprimatu ilara hustuz
        else
                “hutsa” inprimatu
```

Inplementazioa:

```
public static void main(String[] args) {
   Printers p= new Inprimagailuak("c:\\ j95.txt");
   p.datuakProzesatu();
   p.printIlarak();
  }

public class Inprimagailuak extends Object {
 static final int ILARAK = 7;
 Queue<Integer>[] ilarak = new Queue<Integer>[ILARAK+1];
 boolean itzalaldi=false;
 FitxategiIrakurlea file;

  public Inprimagailuak (String f) {
  // ilarak hasieratu
   for (int cc=1;cc<=ILARAK;cc++)
       ilarak[cc]=new Queue();
 hasieratu();
 file=new FitxategiIrakurlea(f);
}

public  void hasieratu() {
 for (int c=1;c<=ILARAK;c++)
  ilarak[c].empty();
}

public void datuakProzesatu() {
 Data d;
 while(file.hasMoreTokens())
    {
     d=file.getTokens();
     if (d.getOperation()=='E') {
      System.out.println("Inpresio "+ d.getPrinter() + "
                                        "+ d.getFileNumber());
      if (Itzalaldi) txertatuLana(6,d.getFileNumber());
       else
           txertatuLana(d.getPrinter(),d.getFileNumber());
     }
```

```
        if (d.getOperation()=='B') {
         System.out.println("Bukaera "+ d.getPrinter() );
         bukatuLana(d.getPrinter());
        }
       if (d.getOperation()=='I') {
        System.out.println("Itzalaldi");
        lanakPasatu();
        itzalaldi=true;
        }
       if (d.getOperation()=='K') {
         itzalaldi=false;
         System.out.println("Sustituzio");
         }
  }
}

  public  void txertatuLana (int inp, int t) {
   ilarak[inp].enqueue(new Integer(t));

   }
   public  void lanakPasatu() {
    for (int c=1;c<=ILARAK-2;c++)
     while (!ilarak[c].isEmpty()){
      Integer lana=ilarak[c].dequeue();
      ilarak[6].enqueue(lana);
    }
    }
   public  void bukatuLana (int inp) {
     ilarak[inp].dequeue();
   }
   public  void printIralak() {

    System.out.println("Ilaren egoera");
    for (int c=1;c<=ILARAK-1;c++) {
     System.out.println(" ");
     System.out.println(c+" ilararen egoera);
     Enumeration lanak=ilarak[c].elements();
     while (lanak.hasMoreElements())
      System.out.print(lanak.nextElement());
    }
   }
}
```

(Ir95)

Espezifikazioa:
--Aurre: F_HASIERA.DAT kaiaren hasierako egoera dauka. Lerro bakoitzak edukiontzi bat eta bere kokapena dauka. F_GARABIA.DAT lerro bat dauka mugimendu bakoitzentzat bai (a)kaiatik itsasontzirantz (C) edo (b) kamioetik kaiara (D). Mugimendu guztiek edukiontzia eta kokalekua daukate. Bi fitxategietan konzeptuak hutsune batengatik banatzen dira. Edukiontzi bakoitza hizki bat eta bi zenbaki osoengatik izendatuko da.
--Pos: F_GGARABIA.RES Lerro bat dauka egin behar diren mugimendu guztiekin. Mugimenduak egiteko 10. kokalekua erabiltzen da laguntzaile bezala, edukiontziak denboraldi batez uzteko.

Datu egitura(k):
19 ilarako pila kokaleku zenbakiagatik indexatuta. Edukiontzientzat(String) pilak erabiliko dira, edukiontziak kokaleku gailurretik ipini eta hartzen baitira.

Algoritmoa:

1.Hasierako egoera lortu
2.Eskaerak exekutatu mugimenduak inprimatuz

1. <u>Hasierako egoera lortu</u>
kokaleku guztientzat
    Bere pila hutsa ipini
1.1. F_HASIERA.DAT fitxategiko edukiontziak ipini.

2. <u>Eskaerak exekutatu mugimenduak inprimatuz</u>
fitxategiak iriki
while (elementuak fitxategiak)
    Eskaera mota, edukiontzi eta kokalekua irakurri
    If (kamioi->kokaleku)
      Kamioitik kokalekura mugitu
      Mugimendua idatzi
    else
      gaineko edukiontziak mugitu kokaleku laguntzailera
      edukiontzia kokalekutik itsasontzira mugitu
      Mugimendua idatzi
      Edukiontziak mugitu laguntzaile kokalekutik hasierako kokalekura
    endIf
endWhile
fitxategiak itxi

1.1. <u>F_HASIERA.DAT fitxategiko edukiontziak ipini</u>
F_INICIAL.DAT iriki
While (elementuak)
    Edukiontzia eta kokalekua irakurri
    Edukiontzia kokalekuan ipini
endWhile
F_INICIAL.DAT itxi

Inplementazioa:

Programa Nagusia

```
public static void main(String[] args) {
    String initData="c:\\DataStructures2\\data\\s95\\f_hasiera.txt";
    String movements="c:\\DataStructures2\\data\\s95\\garabia.txt";
    Portua p=new Portua();
    p. kontainerrakKargatu (initData);
    p.mugimenduakEgin(movements);

  }

public class Portua {
static final int KOKALEKU = 20;
public  Stack<String>[] kok= new Stack<String>[KOKALEKU+1];

  public Portua() {
  // kokalekuak hasieratu
  for (int k=1;k<KOKALEKU;k++) {
     kok[k]=new Stack();
     kok[k].clear();
    }
}
public  void mugimenduakEgin(String f) {
 FitxategiIrakurlea file=new FitxategiIrakurlea(f);
 Data2 d;
   while(file.hasMoreTokens()) {
    d=file.getTokens();

    char op=d.getOp();
    if (op=='D') {
        kok[d.getKok()].push(d.getContainer());
        System.out.println("Kamiotik mugitu  "+d.getKok() +"kokalekura");
       }
     else
      {
       gainetikKendu(d.getContainer(),d.getKok());
       // itsasuntzira mugitu
       cont[d.getKok()].pop();
       System.out.println("Portutik mugitu "+d.getKok()+" itsasuntzira");
       kokalekuanIpini(d.getKok());
      }

 }
}
```

```
private  void gainetikKendu (String b, int c) {
  while ( (!kok[c].isEmpty()) && ( !(kok[c].top()).equals(b)) )
    {
    kok[10].push(kok[c].pop());
    System.out.println("Mugitu "+c+”10 kokalekura");
    }
 }


private void kokalekuanIpini(int c) {
 while (!kok[10].isEmpty())
    {
    System.out.println("Mugitu 10  "+c+” kokalekura”);

    kok[c].push(kok[10].pop());

    }
 }
  public void kontainerrakKargatu(String f) {

   FitxategiIrakurlea2 file=new FitxategiIrakurlea2(f);
   Data d;
     while(file.hasMoreTokens()) {
      d=file.getTokens();
      kok[d.getKok()].push(d.getContainer());
      }
   }

}
```

(Ek96)

Espezifikazioa:
--Pre: Biltegiko kaxa informazioaren fitxategia. Produktu baten kaxak, iraungipen dataz ordenatuta. Biltegitik sartu eta atera diren kaxen fitxategia. Iraungipen data gertuen duen kaxa, lehen ateratzen den kaxa da.
--Pos: Hiru egunetan iraungitzen diren kaxa guztien fitxategia. Bost kaxa baino gutxiago geratzen diren produktuen fitxategia.

Datu egitura(k):
100 elementuko Array-a (bat produktu bakoitzentzat) produktu datu motatakoa. Produktu objektuak, ilara bat eta ilaran dauden elementu kopurua(oso bat) gordezen du. Ilarako elementuak data motatakoak dira(gordetako kaxaren iraungipen data).

Algoritmoa:

```
Egitura hasieratu
1. Egitura jaso fitxategitik
2. mugimenduak prozesatu
3. emaitzak lortu
```

```
1. Egitura jaso fitxategitik
fitxategia iriki
while (elementuak fitxategian)
        kaxa taldea irakurri
        kaxa taldea txertatu
endWhile
fitxategia itxi
```

```
2. mugimenduak prozesatu
fitxategia iriki
while(elementuak fitxategian)
        mugimendua irakurri
        mugimendua egin
endWhile
fitxategia itxi
```

```
3 Emaitzak lortu
Fitxategiak sortu
for Produktu bakoitzentzat
        Zenbatzailea 0-ra hasieratu
        While (kaxak eta iraungipen data<3)
                Zenbatzailea gehitu
                Lehenengo kaxa kendu
                Elementu kopurua txikitu
        endWhile
        if zenbatzailea > 0
                kaxa zenbatzailea idatzi
        end if
        if 5 kaxa baino gutxiago badaude
                eskaera berria idatzi
        end if
endFor
fitxategiak itxi
```

Inplementazioa:

```
public static void main(String[] args) {
    String initialData="c:\\f_hasiera.txt";
    String movements="c:\\f_mugimenduak.txt";
    SuperMarket market=new SuperMarket();
    market.hasierakoDatuakIrakurri(initialData);
    market.mugimenduakExekutatu(movements);
    market.produkuakInprimatu ();
  }

public class SuperMarket {
static final int ILARAK = 7;
static Queue<Integer>[] ilarak =
                                new Queue<Integer>[ILARAK+1];
//Eraikitzailea
  public SuperMarket() {
  // ilarak hasieratu
   for (int cc=1;cc<=ILARAK;cc++)
       ilarak[cc]=new Queue<Integer>();
   hasieratu();
}

public void hasieratu() {
 for (int c=1;c<=ILARAK;c++)
  ilarak[c].clear();
}

public void hasierakoDatuakIrakurri (String f) {
  FitxategiIrakurlea1 file=new FitxategiIrakurlea1(f);
 // eskaera bat tratatu
   Data d;
   while(file.hasMoreTokens()) {
    d=file.getToken();
    ilarak[d.getArt()].enqueue(new Integer(d.getCad()));
   }
 }
```

```
public static void mugimenduakExekutatu (String f){
 FitxategiIrakurlea1 file=new FitxategiIrakurlea1(f);
 Integer i;
 // eskaera bat tratatu
   Data2 d;
   while(file.hasMoreTokens()) {
    d=file.getToken();
    if (d.getOp()=='E')
     addProduct(d.getArt(),d.getCad());
     else
      i=ilarak[d.getArt()].dequeue();
   }
}

public  void produkuakInprimatu() {
 System.out.println("produktuen egoera");
 for (int c=1;c<=ILARAK-1;c++) {
  System.out.println(" ");
  System.out.println(c +"produktuen egoera”);
  Enumeration prods<Integer>=ilarak[c].elements();
  while (prods.hasMoreElements())
     System.out.print(prods.nextElement());
   }
  }
}
```

(S96)
Espezifikazioa:
--Pre: Gutunen informazio fitxategia: Barrutia eta helbidea. Bulegora ailegatu diren ordenean.
--Pos: Gutunen banatze fitxategia. Barruti bakoitzentzat gutun sorta bat dago (200 gehienez). Sortak barruti zenbaki txikienetik handienera ordenatuta daude. Barruti baten gutunak bulegoan sartu diren ordenean gordetzen dira.
-- Beste fitxategi bat, egitura berdinekin, banatu ezin daitezken gutunekin.

Datu egitura(k):
10 elementuko Array-a (bat barruti bakoitzentzat) barruti datu motatakoa. Barruti objektuak, gutun kopurua eta gutunen helbideeen ilara bat dauka.

Algoritmoa:

```
Egitura hasieratu
2. Gutunak banatu
3. Emaitzak lortu
```

```
2. Gutunak banatu
fitxategia iriki
while (elementuak fitxategian)
    helbidea irakurri
    Barrutiaren ilaran helbidea txertatu
    Barrutiaren gutun kopura gehitu
endWhile
fitxategia itxi
```

```
3. Emaitzak lortu
fitxategiak sortu
for (barruti bakoitzentzat)
    Zenbatzailea 0-ra hasieratu
    While (barruti ilara ez hutsa eta
            Zenbatzailea <200)
      Ilararen lehena banaketan ipini
      Ilaratik lehena atera
    endWhile
    (Barruti gutunak-200)gainontzekoak idatzi
    while (barruti ilara ez hutsa)
      Ilararen lehena gainerakoan idatzi
      Ilararen lehena atera
    endWhile
endFor
```

Inplementazioa:

Erabilitako osagarri klaseak

```
 public class Barruti extends Object {
   static final int KARTAK=200;
   Queue<String> ilara;
   int elems;

    public Barruti() {
      ilara=new Queue<String>();
      elems=0;
    }
 }
```

Programa Nagusia

```
public static void main(String[] args) {
    String file="c:\\DataStructures2\\data\\s96\\s96.txt";
    PostOffice po=new PostOffice(file);
    po.gutunakBanatu();
    po.emaitzakLortu();
  }

public class PostOffice {
static final int POSTARIAK = 4;
Barruti barrutiak[]=new Barrutia[POSTARIAK+1];
FitxategiIrakurlea file;
  /**
   * Eraikitzailea
   */
  public PostOffice(String f) {
  file=new FitxategiIrakurlea(f);
  // Barrutiak hasieratu
   for (int cc=1;cc<=BARRUTIAK;cc++)
       barrutiak [cc]=new Barrutia();
  }
  public  void emaitzakLortu() {
  for (int cc=1;cc<=POSTARIAK;cc++) {
    System.out.println(cc+  "postariari  esleitutako  kartak
   "+barrutiak[cc].elems);
    while (!barrutiak[cc].ilara.isEmpty() ){
     String s=barrutiak[cc].ilara.dequeue();
      System.out.println(s);
      }
    }
  }
```

```
  public  void gutunakBanatu() {
     Data d;
     while(file.hasMoreTokens()) {
      d=file.getTokens();

    barrutiak[d.getBarrutia()].ilara.enqueue(d.getPlace());
      barrutiak[d.getBarrutia()].elems++;
      }
  }
}
```

(J97)

Espezifikazioa:

--Pre: HASIERA.TXT fitxategia, mamu datuekin lehentasun ordenean kokatuta izualdiak emateko. Lerro bakoitzak, mamuaren NAN-a, emandako izualdi kopurua eta logela zenbakia dauka.

-- ERRESERBAK.TXT fitxategia, logelen erreserbekin, ostalari iriste ordenean. Lerro bakoitzak, logela zenbakia eta erreserbatutako gauak dauka.

--Pos: Bizitza hobeagora iragan diren mamuen NAN-a, bizitza hobeagora iragan diren mamu gehienen logela zenbakia.

Datu egitura(k):

25 posiozko Array-a (bat logela bakoitzentzat). Elementu bakoitzak bi elementu dauzka:

- Bizitza hobeagora iragan diren mamu zenbatzailea.
- Ilara bat bi elementuekin: mamuaren NAN-a ata emandako izualdi kopurua.

Algoritmoa:

```
Egitura hasieratu
  1. hasierako datuak jaso
  2. Erreserbak kudeatu, irteerak idatziz eta logela maximoa mantenduz
Logela maximoa idatzi
```

1. <u>Hasierako datuak jaso</u>

```
fitxategia iriki
While (elementuak fitxategian)
    Lerroa irakurri
    NAN/izualdiak bikotea txertatu
      emandako logela zenbakian
endWhile
itxategia itxi
```

2. <u>Erreserbak kudeatu, irteerak idatziz eta logela maximo mantenduz</u>

```
fitxategia iriki
Goiburu 1 idatzi pantailan
Hasieratu logela maximoa lehen logelari
While (erreserbak irakurri gabe)
        Erreserba datuak irakurri
        For (erreserbatutako gauengatik eta
                mamuak geratzen badira logela
                 horretan)
                 2.1 Logelaren lehen
                     mamuaren izualdia tratatu
        endFor
endWhile
fitxategia itxi
```

2.1 <u>Logelaren lehen mamuaren izualdia tratatu</u>

```
Logelako ilaratik lehen mamua atera
Bere izualdi kopura gehitu

if (bizitza hobeagora pasatzen bada)
        NAN idatzi
        Logelaren mamu kopurua bizitza
        hobeagoan gehitu
        if (logela maximoa bada)
                logela maximoa eguneratu
        endIf
  else
        logelako ilaran txertatu
endIf
```

Inplementazioa:

Erabilitako osagarri klaseak

```
public class Logela extends Object {
     static final int MAMUAK=10;
     Queue<Mamu> ilara;
     int nManuBizitzaHobe=0;

      public Logela() {
           ilara=new Queue<Mamu>();
     }
}
```

Programa Nagusia

```
public static void main(String[] args) {
   String initialData="c:\\hasiera.txt";
   String bookData="c:\\erreserbak.txt";
   Gaztelu g=new Gaztelu();
   g.hasierakoDatuakKargatu(initialData);
    g.ErreserbakKudeatu(bookData);
    g.logelakMaximoaIdatzi();
  }

public class Gaztelu {
static final int LOGELAK = 25;
Logela logelak[]=new Logela[LOGELAK+1];
  /**
   * Eraikitzailea
   */
  public Gaztelu() {
  // Logelak hasieratu
   for (int log=1;log<=LOGELAK;log++)
       logelak[log]=new Logela();

  }
  public  void logelakMaximoaIdatzi() {
  int nLogMax=0;
   int nMamuLogMax=0;
   for (int log=1;log<=LOGELAK;log++)
       if (logelak[log].nManuBizitzaHobe> nMamuLogMax) {
        nMamuLogMax =logelak[log]. nManuBizitzaHobe;
        nLogMax=log;
        }
      System.out.println("Logela "+ nLogMax+ “da”);
  }
  public  void hasierakoDatuakKargatu (String f) {
  FitxategiIrakurlea file=new FitxategiIrakurlea(f);
  Data d;
while(file.hasMoreTokens()) {
 d=file.getTokens();
 Mamu m=new Mamu(d.getId(),d.getIzualdi());
```

```
      logela[d.getLogela()].ilara.enqueue(m);
      }

   }

public  void erreserbakKudeatu (String f) {
  FitxategiIrakurlea2 file=new FitxategiIrakurlea2(f);
  Data2 d;
  System.out.println("Mamuak bizitza hobeagora");
       while(file.hasMoreTokens()) {
        d=file.getTokens();

        int zLog=d.getLogela();
        int zGauak=d.getGauak();

         while ((zGauak>0) &&
                (!logelak[zLog].ilara.isEmpty() )) {
          (logelak[zLog].ilara.front()).zIzualdi++;
           zGauak--;
         Mamu m=logelak[zLog].ilara.front();
         if (m.zIzualdi==10) {
          System.out.println(m.nan);
              logelak[zLog].ilara.dequeue();
              logelak[zLog].nManuBizitzaHobe++;
              }
          }
        }
      }
  }
```

(S97)
Espezifikazioa:
--Pre: Textu fitxategia. Lerro bakoitzak telefonozko deialdi baten informazioa jasotzen du: deitu den zenbakia, deialdia egin den luzapena eta deialdiaren minutuak. Deialdiak egin diren ordenean agertzen dira.
--Pos: Textu fitxategia. Goiburuak, gehiago deitu den zenbakia (elkarrizketa guztien minutuen batura) eta elkarrizketa guztien minutu kopurua gordezen du
-- Zenbaki horretara egin ziren deialdi guztiak gordetzen dira, egin ziren ordenean mantenduz.

Datu egitura(k):
100 elementuzko Array-a, telefono zenbakiengatik indexatuta. Elementu bakoitzak bi datu gordeko ditu:

- Telefono zenbaki horretara egin diren deialdien minutuen batura
- Ilara bat beste bi datuekin. Deialdia egin zen luzapenetik, eta elkarrizketaren minutuak.

Algoritmoa:

```
Egitura hasieratu
1. Sarrera tratatu egitura eguneratuz
2. Emaitza lortu egituratik
```

```
1.Sarrera tratatu egitura eguneratuz
fitxategia iriki
Denbora maximoa 0-ra hasieratu
While (deialdiak irakurri gabe)
        deialdia irakurri
        deialdia tratatu
        denbora maximoa eguneratu
        telefono maximoa eguneratu
endWhile
fitxategia itxi
```

```
2.Emaitza lortu egituratik
fitxategia sortu
if (denbora maximoa=0)
  Deialdirik ez direla egon inprimatu
else
  Goiburua idatzi
  While (ilara ez hutsa)
        Luzapena eta iraunpena idatzi
        Ilararen lehena ezabatu
  endWhile
endIf
fitxategia itxi
```

Inplementazioa:

Erabilitako osagarri klaseak

```
public class Deialdi {
int luzapena;
int denbora;

 public Deialdi (int pLuzapena, int pDenbora) {
  luzapena= pLuzapena;
  denbora= pDenbora;
  }
}

public class Telefono {
static final int TELEFONOAK=10;
Queue<Deialdi> ilara;
int denb=0;

  public Telefono() {
  ilara=new Queue<Deialdi>();
  }
}
```

Programa Nagusia

```
public static void main(String[] args) {
    String file="c:\\DataStructures2\\data\\s97\\s97.txt";
    Phone p=new Phone(file);
    p.datuakKargatu();
    p.deialdiakProzesatu();
  }

public class Phone extends Object {
static final int TELEFONOAK = 100;
static Telefono telefonoak[]=new Telefono[TELEFONOAK +1];
FitxategiIrakurlea file;
  //Eraikitzailea
  public Phone(String f) {
   file=new FitxategiIrakurlea(f);
   for (int tel=0;tel<= TELEFONOAK;tel++)
       telefonoak[tel]=new Telefono();
  }
```

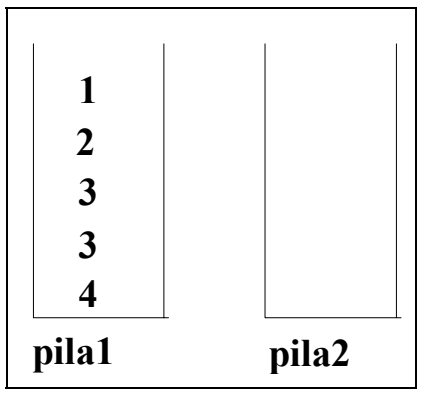
```
public  void datuakKargatu() {
    Data d;
     while(file.hasMoreTokens()) {
      d=file.getTokens();
      Deialdi dei=new Deialdi(d.getLuzapena(),d.getDenb());
      telefonoak[d.getTelephone()].ilara.enqueue(dei);

telefonoak[d.getTelephone()].denb=telefonoak[d.getTelephone()].denb+d.getDenb();
     }
}

 public  void deialdiakProzesatu() {
   int nTelMax=0;
   int nTelDenbMax=0;
   for (int tel=0;tel<= TELEFONOAK;tel++)
       if (telefonoak[tel].denb> nTelDenbMax) {
        nTelDenbMax =telefonoak[tel].denb;
        nTelMax=tel;
        }
       System.out.println("Gehiago deitzen duen telefonoa "+nTelMax);
       System.out.println("Egindako deialdiak");
       Deialdi dei;
       while (!telefonoak[nTelMax].ilara.isEmpty() ) {
        dei=telefonoak[nTelMax].ilara.dequeue();
        System.out.println(dei.luzapena+" "+dei.denbora);
       }
}}
```

(S98)

a) Osoen zerrenda: 1,2,3,3,4:



b) Ez, beti pila1-ean egongo dira zerrendako elementuak. Lehenengoa pilaren gailurrean eta azkenekoa pilaren hondoan.

c) isHutsa() eta ezabatuOrdenatua(int pBalioa) metodoak **diseina e inplementatu**.

```
public boolean isHutsa() {
return pila1.empty();
}
```

EzabatuOrdenatuaAlgoritmoa:

elem baino txikiago diren elementuak pila2-an ipini
pila1-eko *elem* elementuaren agerpenak ezabatu
pila2 pila1-ean hustu

```
public void ezabatuOrdenatua(int pBalioa) {
 Integer elem=new Integer(pBalioa);
 pila2.clear();
 //elem baino txikiago diren elementuak pila2-an ipini
 while ((!pila1.empty()) && ((pila1.top()).compareTo(elem)<0))
    pila2.push(pila1.pop());
  //pila1-eko elem elementuaren agerpenak ezabatu
 while ((!pila1.empty()) && ((pila1.top).compareTo(elem)==0))
  pila1.pop();
 //pila2 pila1-ean hustu
 while (!pila2.empty())
   pila1.push(pila2.pop());
}
```

d) Eragiketa guztiak konplexutasun orden berdina daukate bi kasuetan.

e) Nahiz eta konplesutasun ordena berdina izan bi kasuetan, bi piletan egin behar diren eragiketa kopurua, zerrendan egin behar dena baino handiagoa da kasu gehienetan (ezabatuOrdenatua eragiketan adibidez, egin behar diren eragiketak bikoitzak dira). Errepresentazioa pilen bidez ez du inplementazioa eta ulergarritasuna errezten.