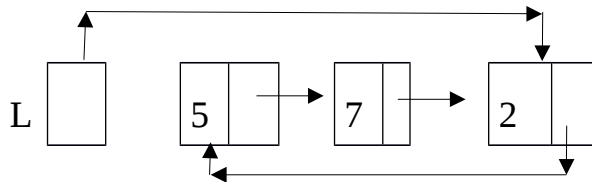


DATU-EGITURAK ETA ALGORITMOAK

EKAINA 2016

1. Zerrenda biderkatu (2,5 puntu)



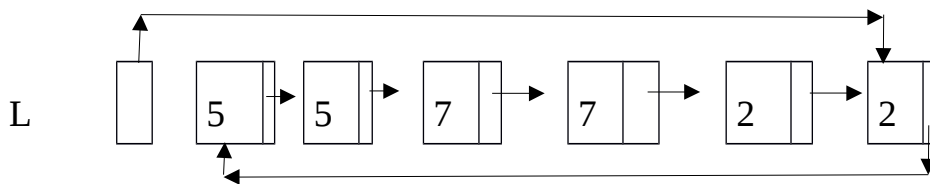
Estekadura zirkularreko zerrenda emanda (zerrendako azken elementuaren atzipenaren bidezkoa), ondoko metodoa kalkulatu nahi da:

```
public Node<T> {
    String data;
    Node<T> next;
}

public class Lista<T> {
    Node<T> last;

    public void biderkatu(Integer n) {
        // aurre: n >= 1
        // post: zerrenda aldatu da, eta hasierako zerrendako elementu
        //       bakoitza "n" aldiz dago
    }
}
```

Aurreko adibideko zerrenda emanda, *biderkatu(2)* deiak ondoko zerrenda bueltatuko luke:



a) *biderkatu* metodoa inplementatu

b) Algoritmoaren kostua kalkulatu, **modu arrazoituan**

2. Blokeen jokoa (2,5 puntu)

Ondoko definizioak emanda:

```
public class Bloke {
    int puntuak;
    int jauzia; // -3 eta +3 arteko balioa
}

public class Jokoa {
    Stack<Bloke>[] taula; // pilen array-a
    public static int ZUTABEKOP = 7;

    public Jokoa() { // eraikitzailea
        taula = (Stack<Bloke>[]) new Stack[ZUTABEKOP];
        for (int i = 0; i <= ZUTABEKOP - 1; i++) {
            taula[i] = new Stack<Bloke>();
        }
        // blokeen pilak ausaz betetzeko kodea
    }

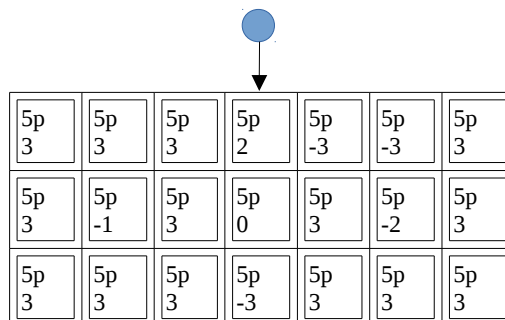
    public int jokatu() {
        // Aurre: jokoa hasieratua izan da (hasierako blokeak sortu dira)
        // Post: partida bat jokatu da. Bola erdiko zutabeen hasten da.
        // Emaita lortutako puntu-kopurua izango da, jokoa gaingitu
        // baldin bada, eta -1 bestela
    }
}
```

Klase horiek joko baterako erabiliko dira. Joko horretan bola bat blokeak “apurtzen” joango da. Bola zutabe baten gainean erortzen denean, orduan zutabe horren gainean dagoen blokea “apurtuko” du. Ondoren, bola apurtutako blokeak adierazten duen posiziora joango da. Norabidea ezkerre edo eskuina izan daiteke, eta jauzia -3 eta +3 bitarteko balioa izango da, norabide horretan pasako diren zutabe-kopurua adieraziz. Jokoa bi arrazoi hauengatik bukatu daiteke:


- Bola blokerik gabeko zutabe batean erortzen da. Kasu honetan jokoa gaingitu egin da.
- Bola zutabeetatik kanpo erortzen baldin bada, orduan jokoa galdu da.

Ondoko irudiek jokoaren adibide bat erakusten dute:

Hasierako egoera (hasteko, erdiko zutabeen eroriko da bola):




Lehenengo blokean erori eta gero, 2 posizioko jauzia egin du eskuinera:




5p 3	5p 3	5p 3		5p -3	5p -3	5p 3
5p 3	5p -1	5p 3	5p 0	5p 3	5p -2	5p 3
5p 3	5p 3	5p 3	5p -3	5p 3	5p 3	5p 3

Blokeak 3 posizioko jauzia emango du ezkerrera.




5p 3	5p 3	5p 3		5p -3		5p 3
5p 3	5p -1	5p 3	5p 0	5p 3	5p -2	5p 3
5p 3	5p 3	5p 3	5p -3	5p 3	5p 3	5p 3

Blokeak 3 posizioko jauzia emango du eskuinera.



5p 3	5p 3			5p -3		5p 3
5p 3	5p -1	5p 3	5p 0	5p 3	5p -2	5p 3
5p 3	5p 3	5p 3	5p -3	5p 3	5p 3	5p 3

Blokeak 2 posizioko jauzia emango du ezkerrera.



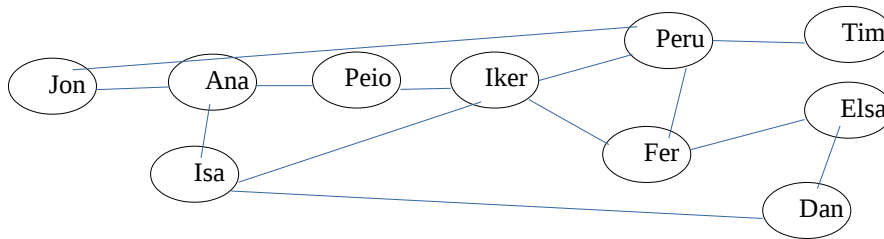
5p 3	5p 3			5p -3		5p 3
5p 3	5p -1	5p 3	5p 0	5p 3		5p 3
5p 3	5p 3	5p 3	5p -3	5p 3	5p 3	5p 3

Eta horrela jokoa amaitu arte.

Jokoa amaitzen denean, programak lortutako puntu-kopurua bueltatuko duen “jokatu” azpiprograma inplementatu behar da.

3. Epidemia (2,5 puntu)

Ondoko grafoak pertsonen arteko erlazioak adierazten ditu:



epidemiarenSimulazioa() metodoa inplementatu nahi dugu pertsona bat, kutsatuta dagoenean, bere ingurukoei ere birusa nola pasako dien neurtu ahal izateko.

Pertsona bat kutsatuta dagoenean, bere kontaktuei kutsatu ahal die gaixotasuna.

Hau *kutsatu()* metodoaren bidez simulatuko da (Pertsona klaseko metodoa, ez da inplementatu behar). Metodo horrek true bueltatuko du birusarekin kontaktuan dagoen pertsona batek gaixotasuna harrapatuko baldin badu, eta false bestela.

Pertsona bat behin birusarekin kontaktuan egonda (gaixotasuna harrapatu dezake ala ez), bigarren batean beste pertsona baten bidez birusarekin kontaktuan egonez gero, ezin izango du gaixotasuna harrapatu.

epidemiarenSimulazioa() algoritmoa idatzi behar da:

```
public class Pertsona {
    String izena;
    boolean kutsatuta; // hasieran false elementu guztientzat
    ArrayList<Pertsona> kontaktuak;

    public void epidemiarenSimulazioa()
        // aurre: pertsona hau kutsatuta dago
        // post: gaixotasuna kutsatu ahal zaie pertsona honen kontaktu guztiei,
        //      eta hauen kontaktuei, ...

    public boolean kutsatu()
        // post: true bueltatzen du birusarekin kontaktuan dagoen pertsona kutsatzen
        //      baldin bada eta false bestela
}
```

4. N hoberenak lortu (2,5 puntu)

M balio positiboen zerrenda dugu ($M = 1.000.000.000$) eta zerrendako N balio handienak lortu nahi dira ($N \ll M$, adibidez, $N = 1.000$).

```
public ArrayList<Integer> nHoberenakLortu(ArrayList<Integer> lista)
```

4 ikasleri galdetuta, soluzio hauek eman dizkigute:

1. soluzioa)

```
emaitza = zerrenda hutsa
errepikatu N aldiz
    bilatu "lista"ko maximoa
    sartu maximoa emaitza zerrendan
    aldatu "lista"ko balio maximoa -1 balioagatik (horrela,
        hurrengoan ez da izango maximoa)
```

2. soluzioa)

```
"lista" beheranzko ordenan ordenatu
hartu lehen N balioak eta bueltatu
```

3. soluzioa)

```
emaitza = zerrenda hutsa
bilaketaZuhaitzBitarra -> sartu "lista"ko M balioak
errepikatu N aldiz
    zuhaitzeko balio handiena bilatu eta ezabatu
    sartu balio handiena emaitza zerrendan
```

4. soluzioa)

```
bilaketaZuhaitzBitarra = zuhaitz hutsa
jatorrizko zerrendako elementu bakoitzeko (M aldiz)
    if zuhaitzaren tamaina < N
        then sartu elementua zuhaitzean
    else if zuhaitzaren tamaina = N and
        zuhaitzeko balio txikiena < uneko elementua
        then ezabatu zuhaitzeko elementu txikiena
        gehitu zuhaitzean uneko elementua
    bueltatu bilaketaZuhaitzBitarra-ko elementuak
```

Hau eskatzen da:

1. Lau algoritmoak ordenatu beraien konplexutasunaren arabera, **algoritmo bakoitzeko bere kostua kalkulatuz, era arrazoituan.**

2. **Implementatu** Javaz algoritmo eraginkorrena.