

# Demo 2 / 25.1

Monisteen numerointia noudattaen (\* = 4 suositeltavaa):

1. Ks. [tehtäviä 4.2 ja 4.3](#). Tehdään järjestämisalgoritmista vielä yksi versio k-kirjaimisille nimille (meillä on lappuja, joissa on jokaisessa yksi sana jossa on k kpl kirjaimia):

0.  $i = k$
1. Laita  $i:n$  kirjaimen mukaan kukin lappu omaan kasaansa nurinpäin (eli on kasa A-kirjaimille, B-kirjaimille, ...)
2.  $i = i - 1$
3. Kerää kasat oikeinpäin päällekkäin siten että ylimmäksi tulee A-kasa
4. jos  $i > 0$ , niin jatka 1.
5. Nimet järjestyksessä.

Jos jokaisessa nimessä olisi 10 kirjainta, niin mikä olisi tämän "jakamisalgoritmin" kompleksisuus (karkeasti ottaen: montako kertaa kutakin lappua pitäisi katsoa)?

2. Kokeile puolitushakua puhelinluetteloon 3:lla keksimälläsi nimellä (joku esiintyy luettelossa, joku ei). Kirjaa ylös "nimien katsomisten" lukumäärä kussakin tapauksessa. Vastaako [tehtävässä 4.10](#) kysytyä kompleksisuutta O-mielessä (tulos olisi siis  $O(\log n)$ , esim. merkintä  $O(n)$  tarkoittaa että operaatioiden määrä on muotoa  $k_1 * n + k_2$ )?

3. Algoritmi "Napoleonin hauta" -pasianssiin sen tarkistamiseksi, käykö tutkittava kortti tiettyyn "pakkaan" ja voiko toisaalta jostakin pakasta ottaa kortin. Ks. [liite](#) ja koeta keksiä taulukkopohjainen toteutus jossa "kaikki" pakat ovat "samanlaisia"! Ohjelmana:

<n:\kurssit\winohj\vc\clx\delphi\korttipe\napoleon.exe>

- 4\*. Keksi ilman lajittelua (siis älä käytä valmista lajittelualgoritmia) toimiva algoritmi joka siirtää pöydälle riviin levitetystä korttipakasta punaiset kortit vasempaan laitaan ja mustat oikeaan laitaan (vaikkei punaisia ja mustia olisi yhtä monta). Vihje: ks. [tehtävä 4.14](#). Käytä "osoittimia". Voit testata algoritmissasi ohjelmalla

<n:\kurssit\winohj\vc\clx\delphi\korttipe\jarjesta.exe>.

- 5\*. Olkoon päiväys muodossa pp.kk. Kirjoita algoritmi, joka lisää päiväystä yhdellä (esim. 25.1 + 1 => 26.1, 31.1 + 1 => 1.2 jne.).TDD: Kirjoita ensin riittävästi esimerkkejä eri vaihtoehdoista.

- 6\*. Kirjoita Java-funktiot kertoman laskemiseksi sekä do-while -silmukalla että while -silmukalla. TDD: Kirjoita ensin kertomat 10 saakka.

$n! = 1*2*3*...n-1*n$ , esim  $3! = 1*2*3 = 6$   
 $0! = 1$

Pöytätestaa (ks. [5.4.2](#)) Java-funktiosi syötöillä 0, 3 ja 6. Päätele myös toimisiko 1:llä.

7. Sijoita 3 korttia kohdan [5.4.5](#) 2-ulotteiseen mallitaulukkoon sekä tee [tehtävä 5.13](#) (muttei mallitehtävän vastauksia).

8\*. "Sievennä" seuraavat ehdot (ks. moniste [5.7.4 Loogiset operaatiot](#)):

Malli: ei ( a < 5 ) sivennetyynä: ( a >= 5 )  
 a) ei hyväksytty joss ( dp < 40% tai vkl < 6 tai summa < 12 )  
 b) ei lennä ulos kapakasta joss (kello < 4.00 ja selvinpäin)  
 c) NOT ( vikoja>90% AND kesto<5 kk)  
 d) ( (kello<7) tai sataa ) ja NOT ( ( kello>=7 ) ja ei sumua )  
 e) kotiin jos ( pimeä ja kylmä ) tai ( pimeä ja pelottaa )

B1. Täydennä liitteenä oleva etu- ja sukunimen vaihtamisohjelma [Etusuku.java](#)B2. Luentomonisteen versiossa 2004 [Luvussa 9.6](#) väitettiin että kahden oliion yhtäsuuruudelta vaaditaan että

Olkoon seuraavassa a1,a2 ja a3 kolme luokan oliota ja b boolean arvo.  
 reflektiivisyys: a1.equals(a1) pitää olla aina tosi  
 symmetrisyys: a1.equals(a2) == a2.equals(a1)  
 transitiivisyys: a1.equals(a2) = b; a2.equals(a3) = b; =>  
 a1.equals(a3) = b;

Näistä transitiivisuutta koskeva väite on väärä. Todista tämä vastaesimerkillä.

G1-2. Tee metodi `pisteet`, jolle viedään parametrinä merkkijono (jonon oltava samaa muotoa kuin yksi `files.txt`:ssä oleva rivi) ja funktio palauttaa reaalilukuna pisteiden lukumäärän (vinkki: Regular Expression ja `java.util.regex.Pattern`-luokka, vaatii vähintään SDK 1.4.1, ks <http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>). `RegExpejä` on kiva opetalla esim. ohjelmalla <http://www.weitz.de/regex-coach/>

Esimerkkejä riveistä ja siitä, mitä palautetaan:

```
teht1.txt = [T1-2:1.5] - oikea muoto          => 1.5
teht1.txt = [T1-2,1.5] - ei :                 => 0.0
teht1.txt = (T1-2:1.5) - väärät sulut        => 0.0
teht1.txt = [T1-2:1,5] - pilkku eikä piste    => 1.0
teht1.txt = T1-2:1.5 - sulut puuttuu         => 0.0
teht1.txt [T1-2:1.5] - ==-merkki puuttuu     => 0.0
teht1.txt = [1-2:1.5] - T-merkki puuttuu     => 0.0
teht1.txt = [B1-2:1.5] - T-merkki puuttuu    => 0.0
```