Tietorakenteet, laskuharjoitus 1, 17.–21.1

Huom: laskarit alkavat jo ensimmäisellä luentoviikolla

1. Tira-kurssilla hyödynnetään usein erilaisia summakaavoja. Tavallinen tapa todistaa summakaavan oikeellisuus on käyttää induktiota.

   (a) Todista induktiolla, että $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

   (b) Todista induktiolla, että $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

2. Tietojenkäsittelijä voi ajatella logaritmia usein seuraavasti: a-kantainen logaritmi $\log_a n$ kertoo, kuinka monta kertaa n pitää jakaa a:lla, ennen kuin päästään lukuun 1. Esimerkiksi $\log_2 8$ on 3, koska 8/2/2/2 on 1. Jos lukuun 1 ei päästä tasajaoilla, logaritmin lopussa on desimaaliosa, joka kuvaa viimeistä vaillinaista jakoa. Esimerkiksi $\log_2 9$ on noin 3,17, koska 9/2/2/2 on 1,125, eli kolmen jaon jälkeen on vielä hieman matkaa lukuun 1, mutta neljäs täysi jako veisi selvästi sen alapuolelle.

   Laske seuraavat logaritmit ilman laskinta:

   (a) $\log_2 4$

   (b) $\log_2 32$

   (c) $\log_3 3$

   (d) $\log_3 81$

   (e) $\log_7 1$

   (f) $\log_7 49$

   (g) $\log_{10} 1000$

   (h) $\log_{10} 10000$

3. "rekursio for dummies"

   Kertaa ohjelmoinnin jatkokurssin materiaalista rekursiota käsittelevä osuus:

   `http://www.cs.helsinki.fi/u/wikla/ohjelmointi/materiaali/VI_OT_kaikenlaista/#6`

   Rekursion hallinta on erittäin tärkeää kurssin kannalta. Jos et osaa tehdä seuraavia tehtäviä omin päin, mene välittömästi TiRa-pajaan ja pyydä ohjaajalta avustusta rekursion oppimisessa!

   Seuraavassa pari yksinkertaista rekursiota käyttävää lämmittelytehtävää. Allaoleviin tehtäviin ei normaalisti käytettäisi rekursiota, eli teemme nämä ainoastaan harjoituksen vuoksi.

   Huom: jos et ymmärrä mitä tehtävässä kysytään tai saa tehtävää tehdyksi, tule kysymään TiRa-pajasta neuvoa!

   (a) Tee metodi, joka tulostaa parametrinaan saamansa määrän tähtiä, eli *-merkkejä. Sen lisäksi funktio kutsuu itseään rekursiivisesti jos parametrin arvo on positiivinen.

   Metodin runko näyttää seuraavalta:

   ```
   private static void tahtia(int lkm) {
      // tulosta lkm tahtea
      // kutsu funktiota rekursiivisesti tulostamaan lkm-1 tahteä
   }
   ```

Käyttöesimerkki:

```
public static void main(String[] args) {
    tahtia(3);
}
```

Ruudulle pitäisi tulostua

```
***
**
*
```

Huom: yksittäinen metodin `tahtia`-komento-osan suoritus siis tulostaa ainoastaan yhden tähtirivin. Eli edellisessä esimerkissä metodi tulee kutsutuksi yhteensä 3 kertaa vaikka main tekeekin ainoastaan yhden kutsun.

(b) Pienellä muutoksella ohjelma saadaan tulostamaan tähdet päinvastaisessa järjestyksessä, eli esim. kutsuttaessa `tahtia(3)`, tulostuu:

```
*
**
***
```

Kokeile mikä muutos saa tämän aikaan. Muutos liittyy rekursiivisen metodikutsun paikkaan, ja onnistuu yhtä koodiriviä siirtämällä. Et tavitse esim. mitään apumuuttujia.

(c) Yksittäinen metodi voi kutsua itseään rekursiivisesti useampaan kertaan. Kirjoita metodin runkoon kaksi rekursiivista kutsua. Laittamalla kutsut sopiviin paikkoihin, saadaan pääohjelmassa kutsumalla tahtia(2) aikaan kuvio:

```
*
**
*
```

Ja kutsumalla tahtia(3) kuvio:

```
*
**
*
***
*
**
*
```

Kokeile mikä muutos saa tämän aikaan.

(d) Muokkaa edellistä ohjelmaa siten, että mukaan tulee staattinen muuttuja jonka avulla voidaan laskea kuinka monennesta rekursiivisesta kutsusta on kysymys. Tulosta jokaisen tähtirivin perään sen aiheuttaneen rekursiivisen kutsun järjestysnumero.

Seuraavassa hahmotelma:

```
static int kutsut;

public static void main(String[] args) {
    kutsut = 1;
    tahtia(3);
}
```

```
private static void tahtia(int lkm) {
    // otetaan talteen monesko kutsu itse ollaan ja kasvatetaan
    // kutsujen yhteenlaskettua lukumäärää
    int kutsunNumero = kutsut++;

    // ...
    // tulosta lkm tahtea ja tulosta kutsunNumero
    // ...
}
```

Edellisen kohdan kutsun tahtia(3) pitäisi näyttää suunilleen seuraavalta (huom. jos kutsut koodissasi rekursiivisesti myös tahtia(0) voi numerointi mennä hieman eri tavalla):

```
* 3
** 2
* 4
*** 1
* 6
** 5
* 7
```

(e) Ymmärrätkö varmuudella sataprosenttisesti miten ohjelmasi etenee? Piirrä miten koodin suoritus etenee tai käy ohjelmasi suoritus askeltaen läpi debuggerilla. Piirtämiseen ja debuggerin käyttöön saat ohjausta tirapajassa.

(f) Kun alat hallita rekursion toimintaperiaatteen, saat pienellä muokkauksella (lisäämällä kolmannen rekursiokutsun) saat ohjelmasi toimimaan esim. seuraavasi:

```
* 3
* 4
* 5
** 2
* 7
* 8
* 9
** 6
* 11
* 12
* 13
** 10
*** 1
```

Tee tämä muutos.

Rekursio on alussa hiukan hämmentävä ohjelmointitekniikka. Älä hermostu tai panikoi jos asia ei auea heti. Opit rekursion vielä kurssina aikana varmasti.

4. Luonnollisen luvun n kertoma $n!$ voidaan määritellä seuraavasti:

$$n! = \begin{cases} 1 & \text{jos } n = 0 \\ n \cdot (n-1)! & \text{jos } n > 0 \end{cases}$$

Esimerkiksi $3! = 3 \cdot 2! = 3 \cdot 2 \cdot 1! = 3 \cdot 2 \cdot 1 \cdot 0! = 3 \cdot 2 \cdot 1 \cdot 1 = 6$.

Toteuta Javalla rekursiivinen funktio, joka laskee luvun kertoman yllä olevalla menetelmällä. Seuraavassa on funktion käyttöesimerkki:

```
private static int kertoma(int luku) {
    // tee funktio tähän
}

public static void main(String[] args) {
    // tulostaa 362880
    System.out.println(kertoma(9));
}
```

5. Jos merkkijonon `TIRA` kääntää ympäri, tuloksena on merkkijono `ARIT`.

   Toteuta Javalla rekursiivinen funktio, joka kääntää merkkijonon ympäri. Seuraavassa on funktion käyttöesimerkki:

```
private static String kaanna(String merkkijono) {
    // tee funktio tähän
}

public static void main(String[] args) {
    // tulostaa ARIT
    System.out.println(kaanna("TIRA"));
}
```

   Vihje: Siirrä ensimmäinen merkki merkkijonon loppuun ja käännä (alkuperäinen) loppuosa rekursiivisesti.

6. Tarkastellaan seuraavaa algoritmia:

   A. Valitaan jokin positiivinen kokonaisluku n.

   B. Jos n on parillinen, jaetaan se kahdella. Jos taas n on pariton, kerrotaan se kolmella ja lisätään tulokseen yksi.

   C. Jos n on yksi, lopetetaan. Muuten palataan kohtaan B.

   Esimerkiksi jos valitaan aluksi $n = 6$, algoritmi etenee seuraavasti:

   - $6/2 = 3$
   - $3 \cdot 3 + 1 = 10$
   - $10/2 = 5$
   - $5 \cdot 3 + 1 = 16$
   - $16/2 = 8$
   - $8/2 = 4$
   - $4/2 = 2$
   - $2/2 = 1$

   Tehtävänä on toteuttaa Java-ohjelma, joka simuloi algoritmia annetulla n:n arvolla. Ohjelman tulostuksen tulee olla seuraavanlainen:

```
Anna luku: 6
6 3 10 5 16 8 4 2 1
```

   (a) Toteuta ohjelma iteratiivisesti (eli ilman rekursiota, joko while- tai for-lauseita hyödyntäen).

   (b) Toteuta ohjelma rekursiivisesti.

   Osaatko sanoa, pysähtyykö ohjelma kaikilla n:n arvoilla?