

## **Opgaver til forelæsningen 30/9-2002**

Vi arbejder med opgaverne 1, 5.7, 5.8 (med tilføjelse) nedenfor; opgave 2 er afleveringsopgave, og vi vil gennemgå opgaveformuleringen ved øvelserne.

**Afleveringsopgave:** Opgave 2 nedenfor; afleveringsfristen er **tirsdag 14 oktober**; opgaven skal afleveres i Hennings postbakke i hus 42.1 eller personligt ved forelæsningen.

**Krav til besvarelsen:** Med mindre I hører andet fra mig, skal I aflevere individuelt, og jeres fulde navn skal fremgå af forsiden. (Det er selvfølgelig tilladt at arbejde sammen 2–3 stykker om at forstå opgaven, men det forventes at I hver især har skrevet ned med jeres egne ord, hvordan I har løst opgaven).

Der er gives ikke ordenskarakter, men jeg forventer at opgaveløsningen er forståelig. Det er ikke nok at aflevere en programtekst med indlejrede kommentarer, men flet det ind i en tekst hvor I forklarer hvordan løsningen er bygget op. Hvor det er relevant skal I også gengive uddata fra testkørsler. **Besvarelser i elektronisk form accepteres ikke.**

Som oplyst andetsteds, stilles der ialt 5 opgaver, hvor I skal aflevere mindst 4. Afleveringsfristerne skal overholdes, og genaflevering er ikke mulig under normale forhold.

### **Opgave 1**

Bogens “bevis” for sætning 5.1 er mildest talt obskurt. Vis ved traditionelle induktionsbeviser følgende:

- 1)  $1 + 2 + \dots + n = n(n+1)/2$
- 2)  $1 + (1+2) + (1+2+3) + \dots + (1+2+\dots+n) = n(n+1)(n+2) / 6$

Jeg *ved* nogen af jer ikke har den matematiske baggrund for at forstå, hvad der menes med et “induktionsbevis”. Derfor skal I løse opgaven grupper, hvor dem som kan løse opgaven forklarer hvad det går ud på til de øvrige.

**Bogens opgave 5.7** (side 177) spg. a og b.

**Bogens opgave 5.8** (side 177) udvidet med følgende spørgsmål. Bestem køretiden udtryk ved store-O for følgende, alternative algoritme, som løser samme problem.

```
public static double power(double x, int n)
{ if (n==0) return 1.0;
  int z = power(n, n/2);
  if (n er et lige tal) return z*z;
  return x*z*z; }
```

NB: Teksten ovenfor har ikke været kørt igennem Java-compileren, så små fejl kan ikke udelukkes.

## Opgave 2

Opgave handler om repræsentation af mængder af tal ved et array udstyret med en tæller; vi skitserer det som en klasse således:

```
public class talmængde{
    public indsæt(int x) {... indsætter x i mængden };
    public boolean med_i(int x)
        {... returnerer true hvis x med i mængden, false ellers};
    private int [] indhold = new int [1000000];
    private int antal = 0
}
```

Problemer hvis arrayet løber fuldt skal ignoreres.

Implementér og aftest klassen i to udgaver

- 1) Tallene indsættes i den rækkefølge de ankommer (dvs. svarende til rækkefølgen af kald af `indsæt`); `med_i` metoden implementeres som sekventielt gennemløb.
- 2) Indholdet af arrayet holdes til enhver tid sorteret. Dette opnås ved at `indsæt` fungerer således:
  - a. først opsøges index hvor det nye tal skal placeres (f.eks. ved binær søgning)
  - b. alle elementer til højre herfor skubbes en position mod højre, så der bliver plads til det nye tal.

Metoden for `med_i` kan implementeres ved binær søgning.

Angiv O for metoderne `med_i` og `indsæt` for begge metoder; hvad siger det om, i hvilke situationer implementation 1 eller 2 er mest velegnede?