Øving 4, algoritmer og datastrukturer

Les oppgaveteksten nøye – altfor mange «har ikke sett» et krav, og må jobbe videre når de ikke får godkjent.

kø, stakk, liste, trær

Her er i alt tre oppgaver. Gjør *to* av dem for å få godkjent. På grupper kan man gjerne løse hver sin del. Vær oppmerksom på at denne oppgaveteksten kan inneholde noe mer arbeid enn det som står i boka, ikke glem å gjøre det.

Innhold

Deloppgave 1 (Josephus problem)

Oppgave 4.3–5 s. 91. Tillegg: Finn dessuten kompleksiteten for programmet dere lager, som funksjon av antall personer (n) og lengden på intervallet (m) mellom dem.

I oppgaven står det «Bruk ei sirkulær liste». Altså ikke andre datastrukturer, som f.eks. tabeller. «arrayList» og «LinkedList» er heller ikke sirkulære.

Et lite tips: hvis listen din har en «slutt», som blir et «spesialtilfelle», har du ikke gjort dette rett. Det blir ikke godkjent.

Andre måter å få denne deloppgaven underkjent:

- bruke indeksering: `tabell[i]`
  - en tabeller ikke ei sirkulær liste
  - vi trenger ikke telle personene her
- bruke oppslag som: `liste.get(i)`
  - vi trenger ikke telle personene
  - `liste.get(i)` er $O(n)$, ødelegger ytelsen.

Deloppgave 2 (matche parenteser, klammer og krøllparenteser)

Oppgave 5–2 s. 109. Programmet skal kunne lese en vanlig kildekodefil, for å sjekke bruken av parenteser, klammer og krøllparenteser. Prøv f.eks. å sjekke programmer fra tidligere øvinger.

Tips: En stakk kan komme godt med her. (Å telle ulike typer paranteser *går ikke* – jeg kan lure alle programmer som er basert på telling.)

> Feil: {[](()) Mangler avsluttende krøllparentes
>
> Feil: {([()])}] En klamme for mye
>
> Feil: (} Disse passer ikke sammen
>
> Feil: [{] } Selv om antallene er korrekte
>
> Rett men komplisert: (){{}[[]][]}{()}
>
> Rett m. kode: int main(){ println("ok");}

Prøv ut grundig. Programmet *skal* kunne håndtere vanlig kildekode, ved å lese *ei fil*. «Hele programmet i en hardkodet streng» er ikke bra nok. Hvis det er brysomt å lese fra fil, trenger dere trening i dette…

Det skal godta alle programmer uten nøstingsfeil, og varsle om alle som har det. (Forenkling: dere trenger ikke ta hensyn til parenteser inni kommentarer eller strengkonstanter)

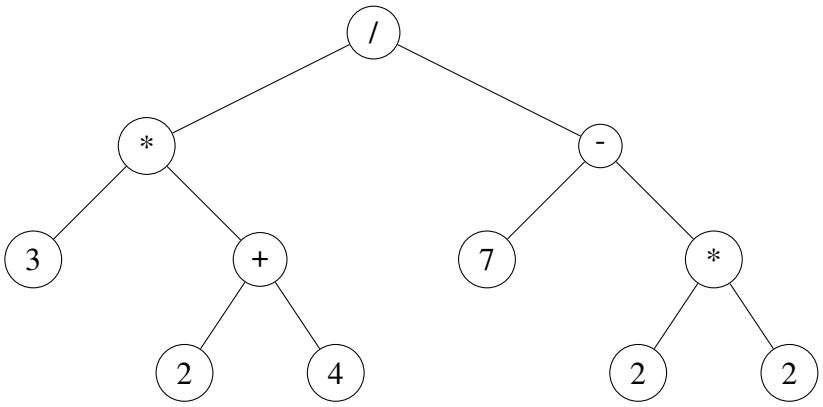
Som alltid, programmet må åpne filer i den mappa det kjører i. For jeg har *ikke* samme mappestruktur som deg. Programmer med mapper jeg ikke har, underkjennes uten videre.

Skriv hele programmet selv. Såpass enkle ting trenger ikke klipp og lim fra nettsider.

Deloppgave 3 (uttrykkstre)

Les oppgaveteksten, det er noe mer å gjøre enn det som står i boka.

Oppgave 6.2–5 på side 122 i læreboka. Lag dessuten et program som kan *beregne* uttrykket som er lagret i et slikt uttrykkstre. Et eksempel:



$\frac{3 \cdot (2+4)}{7-2 \cdot 2} = 6$

Programmet skal altså både skrive ut formelen som ligger i treet, f.eks. "3*(2+4)/(7-(2*2))", og beregne og skrive ut resultatet. (I dette tilfellet 6).

Tips: Alt kan gjøres ved å traversere uttrykkstreet, det er poenget her. Det er *ikke* nødvendig å konvertere til andre datastrukturer, og det er *ikke* nødvendig å lage en parser for strenger som "3*(2+4)/(7-2*2)"