582305 Symbolinen ohjelmointi
3. harjoitus, 10.10.2002

Tehtävä 3.1: Ohjelmoi Scheme-funktio `(latista` l`)`, joka saa syötteenään (mahdollisesti epäaidon) listan l, ja palauttaa sellaisen listan, jonka alkioina on listan l vakioalkiot samassa järjestyksessä, mutta nyt yksitasoisena eikä enää sisäkkäisenä listana.

Esimerkiksi `(latista '(1 (2 . 3) 4 . 5))` = `(1 2 3 4 5)`.

Tehtävä 3.2: Ohjelmoi seuraava Scheme-funktio `(lomita` l_1 l_2`)`:

Syöte l_1 on (aito) lista, jonka alkiot ovat numeroita. Lisäksi nämä alkiot ovat kasvavassa järjestyksessä.
Esimerkiksi l_1 = `(1 1 2 4 5)`.
Myös toinen syöte l_1 on samanlainen järjestetty lukulista.

Tulos on myöskin samanlainen järjestetty lukulista, joka on saatu lomittamalla syötelistojen l_1 ja l_2 alkiot sopivasti.
Esimerkiksi `(lomita '(1 1 2 4 5) '(1 3))` = `(1 1 1 2 3 5)`.

Tehtävä 3.3: Tuttu *lomituslajittelu (merge sort)* etenee seuraavasti:

(a) Jaa lajiteltava aineisto kahteen mahdollisimman yhtäsuureen osaan.

(b) Lajittele nämä osat rekursiivisesti.

(c) Lomita näiden lajittelujen tulokset tehtävän 3.2 tapaan.

Ohjelmoi numerolistojen lomituslajittelu Scheme-funktiona.

Tehtävä 3.4: Tuttu *pikalajittelu (quick sort)* etenee seuraavasti:

(a) Valitse lajiteltavasta aineistosta jakoalkio. Jaa aineisto siten, että edelliseen osaan tulevat korkeintaan ja jälkimmäiseen vähintään jakoalkion suuruiset alkiot.

(b) Lajittele nämä osat rekursiivisesti.

(c) Liitä edellinen lajiteltu osa ja jälkimmäinen lajiteltu peräkkäin.

Ohjelmoi numerolistojen pikalajittelu Scheme-funktiona.

Miksi olisi hyvä idea välttää saman alkion käyttöä useasti jakoalkiona? Miten koodisi voisi tämän tehdä?

Tehtävä 3.5: Tehtävien 3.3 ja 3.4 lajittelumenetelmät muistuttavat kovasti toisiaan:

(a) Jaa lajiteltava aineisto kahteen osaan jollakin periaatteella.

(b) Lajittele nämä osat rekursiivisesti.

(c) Yhdistä nämä lajitellut osat sopivasti.

Ohjelmoi tämä yleinen lajittelumenetelmä korkeamman kertaluvun funktiona, joka ottaa vaiheet (a) ja (c) aliohjelmaparametreina.

Toteuta sitten tehtävien 3.3 ja 3.4 menetelmät funktiollasi.

Onko tällaisesta yleisyydestä mielestäsi hyötyä ohjelmoinnissa?

Tehtävä 3.6: Kalvojen II.8.6 Scheme-kielisellä derivoijalla on (muun muassa) seuraavat ongelmat:

(a) Se sallii kyllä summa- ja tulolausekkeet joissa on enemmän kuin kaksi jäsentä, mutta jättää ylimääräiset huomiotta. Esimerkiksi lausekkeen `(+ x x x)` derivaataksi (muuttujan `x` suhteen) se ehdottaa lauseketta `(+ 1 1)`; kolmas `x` unohtui. Se voisi käsitellä myös nämä lisäjäsenet.

(b) Se tuottaa lausekkeita kuten tuo `(+ 1 1)` jotka voitaisiin sieventää vastaavaan vakioon kuten `2`.

Paranna derivoijaa vastaavasti. Älä kuitenkaan muuta pääfunktiota `deriv` vaan sen apufunktioita: pääfunktion tehtävänä on esittää derivointisääntöjä, apufunktioiden manipuloida lausekkeille valittua esitystä.

(Tehtäviä yhteensä 6 kpl.)