UPPSALA UNIVERSITET
Matematiska institutionen
Salling, Wilander

PROV I MATEMATIK
Algoritmik
26 mars 2008

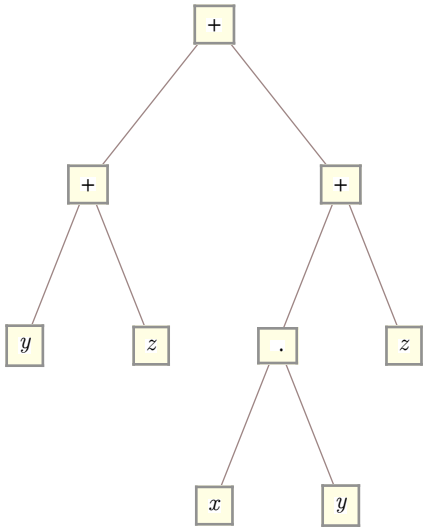
SKRIVTID: 8-12. HJÄLPMEDEL: Inga.
Lösningarna skall åtföljas av förklarande text. För godkänt prov krävs minst 18 p.

---

1. a) (3p) Rita det binära träd vars noder uppräknade i preordning är $+ \; + \; y \;\; z \; + \; \cdot \; x \;\; y \;\; z$.
b) (2p) Räkna upp det binära trädets noder i in- respektive postordning.
c) (1p) Presentera med hjälp av *Add* och *Mult* en funktion som beräknar det aritmetiska uttryck som beskrivs av det binära trädet.

LÖSNING

a)



b) Postordning: $y \;\; z \;\; + \;\; x \;\; y \;\; \cdot \;\; z \;\; + \;\; +$:

Inordning: $y \;\; + \;\; z \;\; + \;\; x \;\; \cdot \;\; y \;\; + \;\; z$

c) $Add(Add(y,\, z),\, Add(Mult(x,\, y),\, z))$

2. (6p) Beskriv vad följande funktioner returnerar. Glöm inte bort att motivera!

a) $\begin{cases} f([\,]) = [\,] \\ f([nod \mid L]) = Om(Lika(Längd(L), 5), nod, f(L)) \end{cases}$

b) $g\big(Öka(n)\big) = h\big(Öka(n), Öka(n)\big)$

$$\begin{cases} h(n, 0) = [\,] \\ h\big(n, Öka(m)\big) = Om\Big(Delbar\big(n, Öka(m)\big), FogaTill\big(h(n, m), Öka(m)\big), h(n, m)\Big) \end{cases}$$

LÖSNING

a) f returnerar det sjätte elementet från slutet för en lista av längd minst 6. Motivering: Om L innehåller exakt 5 element, så är sjätte elementet från slutet i $[nod \mid L]$ lika med nod. Och om L innehåller flera än 5 element, så är det sjätte elementet från slutet i $[nod \mid L]$ samma element som det sjätte elementet från slutet i L.

$$f([a\ b\ c\ d\ e\ f\ g]) \overset{Längd([b\ c\ d\ e\ f\ g])\neq 5}{=} f([b\ c\ d\ e\ f\ g]) \overset{Längd([c\ d\ e\ f\ g])=5}{=} b$$

För kortare listor returnerar f en tom lista. Motivering: Om L är kortare än 5, så finns det inget 6:e element från slutet i $[nod \mid L]$, och inte heller i L. Här är en provkörning:

$$f([a\ b\ c]) \overset{Längd([b\ c])\neq 5}{=} f([b\ c]) \overset{Längd([c])\neq 5}{=} f([c]) \overset{Längd([\,])\neq 5}{=} f([\,]) = [\,]$$

b) g tar input ≥ 1, och returnerar en lista av alla tal som input är delbart med. T.ex. är $g(6) = \{1\ 2\ 3\ 6\}$. Motivering: Vid körning på säg talet $k \geq 1$, så anropas $h(k, k)$. Det som händer då är att h utökar listan $h(k, k-1)$ i höger ände med k, eftersom k är delbar med k. På motsvarande sätt blir listan $h(k, k-1)$ en utökning i höger ände av listan $h(k, k-2)$ om k är delbar med $k-1$, annars blir $h(k, k-1)$ lika med $h(k, k-2)$. O.s.v. Detta upprepas tills h:s högra argument blir lika med 0, då returneras tom lista.

Därför kommer $h(k, k)$ att bli lika med en lista av alla tal som k är delbar med. Provkörning:

$g(6) =$

$$h(6, 6) \overset{6 \text{ är delbar med } 6}{=} [h(6, 5) \mid 6] \overset{6 \text{ är inte delbar med } 5}{=} [h(6, 4) \mid 6] \overset{6 \text{ är inte delbar med } 4}{=} [h(6, 3) \mid 6] \overset{6 \text{ är delbar med } 3}{=}$$
$$[h(6, 2) \mid 3\ 6] \overset{6 \text{ är delbar med } 2}{=} [h(6, 1) \mid 2\ 3\ 6] \overset{6 \text{ är delbar med } 1}{=} [h(6, 0) \mid 1\ 2\ 3\ 6] = [[\,] \mid 1\ 2\ 3\ 6] =$$
$$[1\ 2\ 3\ 6]$$

3. (6p) Skriv en listfunktion som givet en lista av tal returnerar summan av alla tal i listan som är mindre än 10. Till exempel skall funktionen returnera 17 för listan [1 2 3 4 7 10 17 100], liksom för listan [100 12 8 37 9].

LÖSNING

$$\begin{cases} f([\,]) = [\,] \\ f([nod \mid L]) = Om(Mindre(nod, 10), Add(nod, f(L)), f(L)) \end{cases}$$

MOTIVERING:

Att summera de tal i listan $[nod \mid L]$ som är mindre än 10 är lätt om vi redan har summerat alla tal i L som är mindre än 10. Allt som återstår är då att addera nod till det redan summerade ifall $nod<10$.

4. (6p) Tillverka både en stenhögsprocedur och en primitivt rekursiv funktion som givet två naturliga tal m, n beräknar produkten av n stycken på varandra följande tal av vilka det minsta talet är lika med m. T.ex. skall $2\cdot3\cdot4\cdot5\cdot6$ returneras om de två givna talen är 2 och 5. Om $n = 0$, skall talet 1 returneras, oavsett vilket värde m har.
Anmärkning: Du får använda dig av stenhögsproceduren *Multiplicera y med x* respektive den primitivt rekursiva funktionen *Mult*.

LÖSNING

Först en stenhögsprocedur (som lägger resultatet i r).

$r \leftarrow 1$
Sålänge n är icketom {
  Multiplicera r med m
  Öka m
  Minska n}

Sedan en primitivt rekursiv funktion

$$\begin{cases} f(m, 0) = 1 \\ f\big(m, Öka(n)\big) = Mult(f(m, n), Add(m, n)) \end{cases}$$

MOTIVERING:

Att beräkna produkten $2\cdot3\cdot4\cdot5\cdot6$ är lätt om vi redan har beräknat produkten $2\cdot3\cdot4\cdot5$. Det enda som återstår är då att multiplicera (det vi redan har beräknat) med 6.

Allmänt: Att beräkna produkten $\underbrace{m\cdot(m+1)\cdot\ldots\cdot(m+n-1)\cdot(m+n)}_{n+1 \text{ stycken faktorer}}$ är lätt om vi redan har beräknat produkten $\underbrace{m\cdot(m+1)\cdot\ldots\cdot(m+n-1)}_{n \text{ stycken faktorer}}$. Det enda som återstår är då att multiplicera (det vi redan har beräknat) med $m+n$.

5. Givet två godtyckliga listor L_1 och L_2, konstruera
a) (4p) en funktion som returnerar en lista med samtliga gemensamma element till L_1, L_2,
b) (1p) en funktion som avgör ifall L_1och L_2 innehåller något gemensamt element.
ANM. Du får använda dig av funktionen *Tillhör*(x, L) som avgör ifall x tillhör listan L.

LÖSNING

a)

Snittet([ ], Y) = [ ]

Snittet([*nod* | X], Y) =
  Om(*Och*(*Tillhör*(*nod*, Y), *Icke*(*Tillhör*(*nod*, X))),
    FogaIn(*nod*, *Snittet*(X, Y)),
    Snittet(X, Y))

b) *InnehållerGemensamtElement*(X, Y) = *Icke*(*Tom*(*Snittet*(X, Y)))

där *Tom*([ ]) = 1 och *Tom*([*nod* | X]) = 0.

MOTIVERING:

Det är lätt att skapa en lista med de gemensamma elementen till listan [*nod* | X] och listan Y, om vi redan har en lista innehållande de gemensamma elementen till X och Y. Ty då behöver vi bara utöka den redan skapade listan med elementet *nod* ifall *nod* förekommer i listan Y men inte i listan X.

6. (5p) Skriv en funktion som givet ett binärt träd T och en atom x, returnerar en lista av fädrerna till samtliga x-förekomster i T. Du får använda dig av funktionen NivåEtt(T) som returnerar en lista av T:s noder på nivå 1, dvs. nivån under rotnivån.

LÖSNING

Fädrer(x, [ ]) = [ ]
Fädrer(x, [*rot vä hö*]) =
  Om(*Tillhör*(x, *NivåEtt*([*rot vä hö*])),
    FogaIn(*rot*, *FogaSamman*(*Fädrer*(x, *vä*), *Fädrer*(x, *hö*))),
    FogaSamman(*Fädrer*(x, *vä*), *Fädrer*(x, *hö*))
)

MOTIVERING:

Om vi redan har listor med fädren till x-förekomsterna i de binära underträden *vä* respektive *hö*, så kan vi foga samman dessa två listor till en lista. Sedan behöver vi bara utöka det sammanfogade resultatet med *rot* ifall *rot* är far till någon x-förekomst, dvs om x förekommer på nivå 1.

7. (6p) Säg att ett träd av tal är *avtagande* om varje nod i trädet har egenskapen att skogen nedanför nämnda nod saknar större noder. Skriv en funktion som avgör ifall ett träd är avtagande.

LÖSNING

$AvtagandeTräd([rot]) = 1$

$AvtagandeTräd([rot \mid skog]) =$

$\quad Och(AvtagandeSkog(skog),\ Icke(Större(NivåEtt[[rot \mid skog]],\ rot)))$

$AvtagandeSkog([\,]) = 1$

$AvtagandeSkog([träd \mid skog]) = Och[AvtagandeTräd(träd),\ AvtagandeSkog[skog]]$

$Större([\,],\ rot) = 0$

$Större([nod \mid L],\ rot) = Eller(nod > rot,\ \ Större(L,\ rot))$

MOTIVERING:

Ett träd som bara innehåller en enda nod är förstås avtagande. Och ett större träd är avtagande om varje träd i skogen under roten också är avtagande och ingen nod på nivån under roten är större än roten.