Salling, 070 – 6527523
TID : 9 – 13 HJÄLPMEDEL : Inga
BETYGSGRÄNSER
G –18p, VG–28p

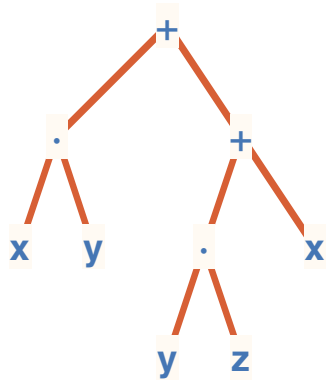
PROV I MATEMATIK
ALGORITMIK 5 p
26 APRIL 2002

Ge klara och kortfattade motiveringar till dina svar !

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

LÖSNING

a)



b) Preordning: $+\ \cdot\ x\ y\ +\ \cdot\ y\ z\ x$:
Inordning: $x\ \cdot\ y\ +\ y\ \cdot\ z\ +\ x$

c) *Add*(*Mult*(x, y), *Add*(*Mult*(y, z), x))

2. Betrakta

```
Evenon x i r =
    Töm y
    Töm z
    Sålänge x > z  {
                Öka y
                Addera y och y i  z }
    x = z? i r
```

a) Vilket innehåll kommer *r* att få efter körning med $x = 0, 1, 2, 3, 4$. (Redovisa Dina provkörningar.) (3p)
b) Konstruera en primitivt rekursiv funktion som beräknar samma sak . (4p)
ANM. Det räcker ej att skriva namnet på en känd funktion. En "konstruktion" efterfrågas.

LÖSNING

a) *r* får innehåll 1,0,1,0,1. Se provkörningarna nedanför:

x	y	z	x > z		r
0	0	0	Falskt		
0	0	0		z = x	1

x	y	z	x > z		r
1	0	0	Sant		
1	1	2	Falskt		
1	1	2		x < z	0

x	y	z	x > z		r
2	0	0	Sant		
2	1	2	Falskt		
2	1	2		z = x	1

x	y	z	x > z		r
3	0	0	Sant		
3	1	2	Sant		
3	2	4	Falskt		
3	2	4		x < z	0

x	y	z	x > z		r
4	0	0	Sant		
4	1	2	Sant		
4	2	4	Falskt		
4	2	4		z = x	1

b) Följande primitivt rekursivt konstruerade funktion beräknar samma sak:

$$Jämn?(0) = 1$$
$$Jämn?(Öka(x)) = Icke(Jämn?(x))$$

3. Beskriv vad följande funktion returnerar? (Glöm inte bort att motivera ditt svar!)

$$
\begin{aligned}
&One([rot\ vä\ hö]) = \\
&\quad Om(Och(Tom?(vä), Tom?(hö)), 0, \\
&\qquad Om(Eller(Tom?(vä), Tom?(hö)), 1, \\
&\qquad\quad Eller(One(vä), One(hö)))))
\end{aligned}
$$

där $Tom?(L) = Noll?(Längd(L))$ (5p)

LÖSNING

Funktionen undersöker om ett icketomt binärt träd innehåller någon nod som har exakt **ett** barn. Annorlunda uttryckt: om någon nod i det binära trädet har exakt **ett** underträd kopplat till sig. Funktionen returnerar 1 om trädet har någon sådan nod, 0 annars.
MOTIVERING: Om de två underträd som hör till inputträdets rotnod är tomma båda två, så returneras 0 direkt. Om exakt ett av dem är icketomt, så returneras 1 "direkt". Annars, dvs om rotnodens underträd är icketomma båda två, så skickas de två underträden in för diagnos, osv. …

4. Givet en lista L innehållande ett godtyckligt antal listor, konstruera en funktion f som sammanfogar alla listorna i L och returnerar sammanfogningen. Tex så att

$f([[a\ b]\ [b\ [b]\ a]\ [\,]\ [c\ b\ a]]) = [a\ b\ b\ [b]\ a\ c\ b\ a]$ (7p)

LÖSNING

Idé: Om alla listor utom den första redan har sammanfogats så behöver man bara foga samman den första listan med resultatet av de redan sammanfogade.

$$
\begin{cases}
f([\,]) = [\,] \\
f([förstalistan \mid L]) = FogaSamman(förstalistan, f[L])
\end{cases}
$$

5. Konstruera en funktion som för ett godtyckligt träd avgör ifall det innehåller någon nod som har exakt två barn (dvs som i grafen är förbunden nedåt med exakt två noder). (7p)

LÖSNING

Lösningen bygger på följande idé:
Ett träd har den sökta egenskapen omm
rotnoden har exakt två barn (dvs om den lista som representerar skogen under rotnoden har längd två) eller om skogen under rotnoden innehåller något träd som har den sökta egenskapen.

```
HarSöktaEgenskapen?([rot | skog] ) =
  Eller (Två?(Längd (skog)),
       InnehållerNågotTrädSomHarSöktaEgenskapen ?(skog))

IInnehållerNågotTrädSomHarSöktaEgenskapen ?([]) = 0
InnehållerNågotTrädSomHarSöktaEgenskapen ?([träd | skog]) =
  Eller (HarSöktaEgenskapen?(träd),
       InnehållerNågotTrädSomHarSöktaEgenskapen ?(skog))
```

6. Uppskatta tidsåtgången, mätt i stora ordo, för var och en av nedanstående två funktioner *g* och *h*. Vi antar att hjälpfunktionerna *Add* och *Mult* var och en har konstant tidsåtgång. Vidare antar vi
(i) att *h* anropas med två lika långa atomära listor i sina två argument, säg att de har längden *n*. Tex kan en beräkning i det fall att $n = 3$ inledas med anropet
$h([2\ 3\ 5],\ [5\ 1\ 2])$.
(ii) att *g* anropas med en **lista av listor** i varje argument, där de senare listorna är atomära och lika långa som antalet listor, säg av längd *n*. Om tex $n = 3$ kan en beräkning inledas med anropet $g([[2\ 3\ 4]\ [1\ 2\ 2]\ [4\ 0\ 1]],\ [[5\ 1\ 2]\ [1\ 1\ 0]\ [3\ 1\ 4]])$. (7p)

LEDNING Mät inputs storlek med det *n* som nämns i texten ovanför.

a)
$$\begin{cases} h([\,],\ B) = 0 \\ h([a \mid A],\ [b \mid B]) = Add(Mult(a, b),\ h(A, B)) \end{cases}$$

b)
$$\begin{cases} g([\,],\ B) = [\,] \\ g([a \mid A],\ B) = FogaIn(h(a, B),\ g(A, B)) \end{cases}$$

LÖSNING

a) $h = O(n)$.
MOTIVERING: Eftersom hjälpfunktionerna *Add* och *Mukt* antas ha konstant tidsåtgång, följer att tidsåtgången för additionen och multiplikationen ges av $O(1)$, och då argumentlistornas längder minskar med en enhet i rekursionssteget följer att *h*:s tidsåtgång ges av

$$\text{Tid}_h(0) = O(1)$$
$$\text{Tid}_h(n) = O(1) + \text{Tid}_h(n-1)$$

Härav,

$$\text{Tid}_h(n) = 1 + \text{Tid}_h(n-1)$$
$$= 1 + 1 + \text{Tid}_h(n-2) = 1 + 1 + 1 + \text{Tid}_h(n-3) = \ldots = n + \text{Tid}_h(n-n) = n + 1 = O(n)$$

b) Tyvärr råkade koden för *g* bli något felaktig i uppgiftstexten. (Jag ber om ursäkt för detta.) Den korrekta koden skulle se ut som nedanför.

$$\begin{cases} g([\,],\ B) = [\,] \\ g([a \mid A],\ [b \mid B]) = FogaIn(h(a, b),\ g(A, B)) \end{cases}$$

$g = O(n^2)$.
MOTIVERING: På motsvarande sätt som för *h* följer för *g* att

$$\begin{aligned}
&\mathrm{Tid}_g(n) = 1 + \mathrm{Tid}_h(n) + \mathrm{Tid}_g(n-1)\\
&= 1 + 1 + \mathrm{Tid}_h(n) + \mathrm{Tid}_h(n) + \mathrm{Tid}_g(n-2)\\
&= 1 + 1 + 1 + \mathrm{Tid}_h(n) + \mathrm{Tid}_h(n) + \mathrm{Tid}_h(n) + \mathrm{Tid}_g(n-3)\\
&= 3 + 3\,\mathrm{Tid}_h(n) + \mathrm{Tid}_g(n-3) = \ldots\\
&= n + n\,\mathrm{Tid}_h(n) + \mathrm{Tid}_g(n-n) = n + n\cdot n + \mathrm{Tid}_g(0) = n + n^2 + 1 = O(n^2)
\end{aligned}$$

(Vi räknar som vanligt med att basfunktionen *FogaIn* har konstant tidsåtgång.)

ANM. Om storleken på *g*:s input mäts med antalet atomer i stället för med listornas längder, blir tidsåtgången linjär:
Låt *m* beteckna antalet atomer i ett av *g*:s argument vid anropet (tex antalet atomer i $[a \mid A]$. Då är $m = n^2$ där *n* precis som förut betecknar längden av $[a \mid A]$.
Härav,

$$\begin{aligned}
&\mathrm{Tid}_g(m) = \mathrm{Tid}_g(n^2) = 1 + \mathrm{Tid}_h(n) + \mathrm{Tid}_g(n^2 - n)\\
&= 1 + 1 + \mathrm{Tid}_h(n) + \mathrm{Tid}_h(n) + \mathrm{Tid}_g(n^2 - 2n)\\
&= 1 + 1 + 1 + \mathrm{Tid}_h(n) + \mathrm{Tid}_h(n) + \mathrm{Tid}_h(n) + \mathrm{Tid}_g(n^2 - 3n)\\
&= 3 + 3\,\mathrm{Tid}_h(n) + \mathrm{Tid}_g(n^2 - 3n) = \ldots\\
&= n + n\,\mathrm{Tid}_h(n) + \mathrm{Tid}_g(n^2 - n\cdot n) = n + n\cdot n + \mathrm{Tid}_g(0) = n + n^2 + 1 = O(n^2)\\
&= O(m)
\end{aligned}$$