Lösningsförslag för Tenta i Programmeringsparadigm 2014-03-13 09.00-12.00

---

1 Prolog

1 (a) Y = 3 respektive Y = 4.
(b) L = [1,3,5] respektive L = [1,3,6].

2 En möjlig användning av snitt är för att implementera negation.

```
man(peter).
man(henry).

woman(X) :-
 man(X),
 !,
 fail.
woman(X).
```

Här är snittet nödvändigt för att implementera negation: utan snittet kommer `woman(X)` alltid lyckas, därför att första regeln kommer alltid misslyckas, och backtrackingen kommer leda till att andra regeln kollas (vilket inte vore fallet med snittet).

(Det finns flera andra möjliga användningar av röda snitt, t.ex. för att undvika oändlig rekursion.)

3 (a)
```
height(leaf, 0).
height(branch(T1, T2), N) :-
  height(T1, N1),
  height(T2, N2),
  max(N1, N2, M),
  N is M+1.
```

(b)
```
complete(leaf).
complete(branch(T1, T2)) :-
  complete(T1),
  complete(T2),
  height(T1, N),
  height(T2, N).
```

---

2 Haskell

4 (a) Två möjliga lösningar:

```
lengthEqThree [_,_,_] = True
lengthEqThree _ = False
```

eller

```
lengthEqThree lst = length lst == 3
```

(b)
```
lengthEqThree :: Num a => [a] -> Bool
lengthEqThree [_,_,_] = True
lengthEqThree _ = False
```

5 I samtliga deluppgifter här kan man för att summera elementen i en lista av längd 3 antingen använda den inbyggda `sum`-funktionen (om man kom ihåg att den finns), eller skriva en egen summerings-funktion på t.ex. det här viset

```
summera [] = 0
summera (f:r) = f + (summera r)
```

eller något mer direkt i stil med `(list !! 0) + (list !! 1) + (list !! 2)`.

Exempellösningarna nedan använder den inbyggda `sum`-funktionen.

(a)
```
sumsOfThrees [] = []
sumsOfThrees (first:rest)
  | lengthEqThree first = (sum first) : (sumOfThree rest)
  | otherwise = sumOfThree rest
```

(b)
```
sumsOfThrees lista = map sum (filter lengthEqThree lista)
```

(c)
```
sumsOfThrees lista = [sum x | x <- lista, lengthEqThree x]
```

(d) Vid svansrekursion bygger man inte på stacken med väntande operationer för att då basfallet nås utföra dessa som den rent rekursiva idén använder. I förslaget nedan används en inre funktion som tar lista som ska behandlas och en tom lista där den nya listan byggs upp.

Svansrekursiv idé

```
sumsOfThrees [[1,2,3], [4,5], [6,7,8]]
    sumsOfThreesInner [[1,2,3], [4,5], [6,7,8]] []
    sumsOfThreesInner [[4,5], [6,7,8]] [6]
    sumsOfThreesInner [[6,7,8]] [6]
    sumsOfThreesInner [] [6,21]
[6 , 21]
```

Rekursiv idé

```
sumsOfThrees [[1,2,3], [4,5], [6,7,8]]
    sumsOfThrees [[1,2,3], [4,5], [6,7,8]]
    6 : sumOfThree [[4,5], [6,7,8]]         :-operatorn kan inte utföras än!
    6 : sumOfThree [[6,7,8]]
    6 : 21 : sumOfThree []                  Först nu kan :-operatorn utföras
    6 : 21 : []
    6 : [21]
    [6 , 21]
[6 , 21]
```

6 (a)
```
data Vectors = V1 Double
             | V2 Double Double
             | V3 Double Double Double
               deriving (Eq)
```

---

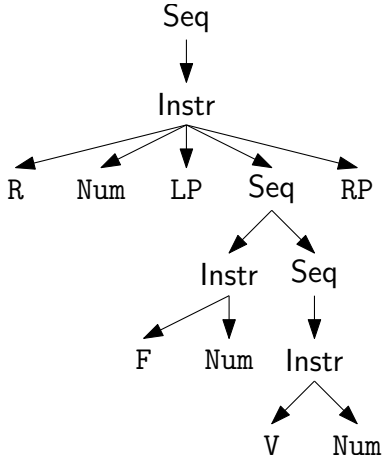
3 Syntax

7 Mening 1: A-1, B-2, C-5
Mening 2: A-4, B-3, C-6

8 (a) En möjlig lösning, med icke-slutsymboler Instr och Seq för instruktioner respektive sekvenser av instruktioner. Start-symbol är Seq.

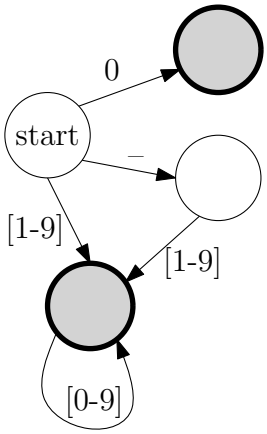
$$\begin{aligned} \text{Instr} &\rightarrow \texttt{V Num} \mid \texttt{H Num} \mid \texttt{F Num} \mid \texttt{R Num LP Seq RP} \\ \text{Seq} &\rightarrow \text{Instr Seq} \mid \text{Instr} \end{aligned}$$

(b) Lösning för grammatiken ovan:



9 (a) Möjlig lösning:
`0|-?[1-9][0-9]*`

(b) Möjlig lösning, de två fetmarkerade gråfyllda tillstånden är accepterande:



(Övergångarna märkta [1-9] är korthand för 9 stycken övergångar märkta $1, 2, 3, \ldots, 9$, och liknande för [0-9])

(c) Möjlig lösning. Startsymbol är Num.

$$\begin{aligned} \text{Num} &\rightarrow \text{Zero} \mid \text{Pos} \mid \text{Neg} \\ \text{Zero} &\rightarrow \texttt{0} \\ \text{OneToNine} &\rightarrow \texttt{1}|\texttt{2}|\texttt{3}|\texttt{4}|\texttt{5}|\texttt{6}|\texttt{7}|\texttt{8}|\texttt{9} \\ \text{Digit} &\rightarrow \text{Zero} \mid \text{OneToNine} \\ \text{Digits} &\rightarrow \text{Digit Digits} \mid \epsilon \\ \text{Pos} &\rightarrow \text{OneToNine Digits} \\ \text{Neg} &\rightarrow \texttt{-}\ \text{Pos} \end{aligned}$$

(mellanslag mellan namn på symboler t.ex. Digit Digits är bara för läsbarhet)