Tenta i Programmeringsparadigm 2015-04-10 08.00-11.00

Inga hjälpmedel är tillåtna. Skriv inga svar på blanketten. Varje del är på 20 poäng. För att bli godkänd på en del krävs 12 poäng på den delen. **Du behöver inte skriva de delar där du redan är godkänd på kontrollskrivningen eller ordinarie tentan.**

Del 1: Funktionell programmering

1. (12p)

   Betrakta följande Haskell-kod.

   ```
   help = (\x y -> x*y)

   lista = [(help 2), (help 0.5), (help 1)]

   funk1 [] _ = []
   funk1 (x:xs) indata = (x indata) : (funk1 xs indata)
   ```

   (a) Förklara i detalj vad som händer vid anropet `(help 2)`. Dvs, där det är möjligt/tillämpbart förklarar du vilka värden som tilldelas till vad, vilka operationer/funktioner som utförs samt vad resultatet av anropet är. *(3p)*

   (b) Vad är typsignaturen för `lista`? *(2p)*

   (c) Vad blir resultatet av anropet `funk1 lista 6.0`? *(2p)*

   (d) Skriv om `funk1` så att den använder listomfattning (list comprehension). *(3p)*

   (e) Konceptet *currying* används i den givna koden men var? *(2p)*

2. (8p)

   Betrakta följande Haskell-kod.

   ```
   data Par a = EttPar a a
                deriving (Eq, Show)

   likaPar par1 par2 = ((fram par1) == (fram par2)) && ((bak par1)==(bak par2))
   ```

   (a) I den givna koden används typvariabeln `a`. Vilket är huvudskälet till att man gör det och vad kallas det? *(2p)*

   (b) Skriv funktionerna `fram` och `bak` så att `likaPar` fungerar. *(2p)*

   (c) Ge ett exempel på ett konkret anrop till `likaPar` som ger resultatet `True`. *(2p)*

   (d) Ange hur man skulle ändra koden ovan för att göra så att funktionen `likaPar` endast fungerar för par av heltal. *(2p)*

Del 2: Logikprogrammering

3. **Kontrollflöde, Backtracking** (6p)

   Betrakta det inbyggda medlemspredikatet för listor, vilket kan definieras enligt följande:

   ```
   member(H, [H | _]).
   member(X, [_ | T]) :- member(X, T).
   ```

   Beskriv kontrollflödet samt unifieringarna steg för steg vid evalueringen av frågan

   ```
   ?- member(5, L), member(-3, L), member(8, L).
   ```

4. **Snitt** (4p)

   Snitt är en kontrollflödeskonstruktion som tillåter oss att skära bort onödig backtracking och därmed göra koden mer effektiv. Står detta i konflikt med programmeringstekniken som använder backtracking för att generera fler eller alla lösningar till ett problem? Motivera ditt svar, gärna med hjälp av exempelpredikat.

5. **Rekursion, Träd** (10p)

   Betrakta datastrukturen inkompletta binära träd, dvs binära träd där ena delträdet (vänstra eller högra) av en intern nod kan saknas.

   (a) Ge en BNF-definition för datastrukturen, som ska lagra data i varje nod. *(2p)*

   (b) Vi säger att ett sådant träd är sorterat om det för varje nod i trädet gäller att all data i det vänstra delträdet är mindre än datat i noden, och att all data i det högra delträdet är större än datat i noden.

   Definiera ett predikat `insel(X, T, TX)` som är sant om och endast om `TX` är sorterade trädet `T` med elementet `X` tillagt.

   Du kan anta att trädet `T` är sorterat, ditt predikat behöver inte hantera indata där `T` inte är sorterat. *(6p)*

   (c) Vad blir svaret på frågan:

   ```
   ?- insel(5, leaf(3), T1), insel(-2, T1, T2), insel(3, T2, T3).
   ```

   (2p)

Del 3: Formella språk och syntaxanalys

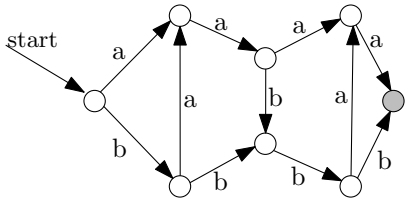
6. (5p)

   (a) Förklara de tre begreppen *sträng*, *språk*, och *språkklass*, och hur de relaterar till varandra. *(3p)*

   (b) *Rekursiv medåkning* är en algoritm/teknik som ibland används inom syntaxanalys. Vad använder man den till, och hur fungerar den (kortfattat)? *(2p)*

7. (4p)

   Betrakta följande DFA.



   Vilka av följande påståenden är korrekta? Motivera dina svar – rätt svar med felaktig eller saknad motivering ger 0 poäng!

   (a) Automaten accepterar oändligt många strängar. *(2p)*

   (b) Det finns ett reguljärt uttryck som beskriver samma språk. *(2p)*

8. (11p)

   I den här uppgiften ska vi arbeta med trollformler. Trollformler är strängar (över tecknen `a-z`) som består av en inledande sträng x (möjligen tom), en huvuddel y (av längd minst 2), en mellandel z (av längd minst 1) och slutligen huvuddelen y upprepad (samma sträng en gång till). Tre klassiska exempel är `hokuspokus` ($x = \texttt{h}, y = \texttt{okus}, z = \texttt{p}$), `abrakadabra` ($x$ är tom, $y = \texttt{abra}, z = \texttt{kad}$), och `simsalabim`, men även moderna påfund som `xzbhfgbhf` ($x = \texttt{xz}, y = \texttt{bhf}, z = \texttt{g}$).

   I den uråldriga trollformelsboken "Syntacto magico paradigmus" hittar man följande kontextfria grammatik för trollformler:

$$\begin{aligned}
\text{Formel} &\rightarrow \text{Inledning Huvud Mellan Huvud} \\
\text{Inledning} &\rightarrow \text{Bokstaver} \\
\text{Huvud} &\rightarrow \text{Bokstav Bokstav Bokstaver} \\
\text{Mellan} &\rightarrow \text{Bokstav Bokstaver} \\
\text{Bokstaver} &\rightarrow \epsilon \mid \text{Bokstav Bokstaver} \\
\text{Bokstav} &\rightarrow \texttt{a} \mid \texttt{b} \mid \texttt{c} \mid \texttt{d} \mid \texttt{e} \mid \texttt{f} \mid \texttt{g} \mid \texttt{h} \mid \texttt{i} \mid \texttt{j} \mid \texttt{k} \mid \texttt{l} \mid \texttt{m} \mid \texttt{n} \mid \texttt{o} \mid \texttt{p} \mid \texttt{q} \mid \texttt{r} \mid \texttt{s} \mid \texttt{t} \mid \texttt{u} \mid \texttt{v} \mid \texttt{w} \mid \texttt{x} \mid \texttt{y} \mid \texttt{z}
\end{aligned}$$

   (a) Är detta en korrekt grammatik för trollformler (enligt beskrivningen ovan)? Motivera ditt svar. *(2p)*

   (b) Är den givna grammatiken tvetydig? Motivera ditt svar. *(2p)*

   (c) Rita ett syntaxträd för härledningen av strängen `simsalabim` enligt den givna grammatiken. *(3p)*

   (d) Är språket med syntaktiskt korrekta trollformler reguljärt? Motivera ditt svar. *(4p)*

   Ledtråd till (d): ett ekvivalent sätt att definiera vad som är en giltig trollformel är att kräva att huvuddelen y ska ha längd exakt 2 istället för minst 2. (Du får i ditt svar använda denna alternativa definition, men för full poäng ska du isåfall bevisa att detta är ekvivalent.)