Tenta i Programmeringsparadigm 2016-01-15 14.00-17.00

Inga hjälpmedel är tillåtna. Skriv inga svar på blanketten. Bonuspoäng från Inet-labben hösten 2015 kommer automatiskt att tillgodoräknas på den del av tentan där de gör mest nytta. 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.**

Del 1: Funktionell programmering

1. Givet följande kod:

```
char2bin [] = []
char2bin (t:text)
  | svTecken t = t : char2bin text
  | otherwise = char2bin text
```

   Du kan anta att funktionen `svTecken` tar indata av typen `Char` och returnerar `True` om indata är '`å`', '`ä`', '`ö`', '`Å`', '`Ä`' eller '`Ö`' annars `False`.

   (a) Ange noggrannt beräkningsstegen för anropet `char2bin "Gröt"`. *(2p)*

   (b) Funktionella språk använder sig inte av variabler som t.ex. imperativa och objektorienterade språk gör. Förklara *utifrån den givna koden* hur man löser detta i det funktionella paradigmet. *(2p)*

   (c) Använder den givna koden den svansrekursiva idéen? Motivera ditt svar tydligt och noggrannt. *(1p)*

   (d) *Om* den givna funktionen använder sig av den svansrekursiva idéen, skriv om den så att den är rekursiv *utan* att använda svansrekursion, eller vice versa (om funktionen inte använder sig av svansrekursion, skriv om den så att den gör det).

   Funktionaliteten ska bibehållas! Du får *inte* använda dig av listomfattning eller inbyggda funktioner som `map` eller `filter`. *(4p)*

   (e) Skriv en ny funktion med samma funktionalitet som använder sig av listomfattning och *inte* använder sig av rekursion eller inbyggda funktioner som `map` eller `filter`. *(3p)*

2. Givet följande kod:

```
xs = [1,2,3]::[Float]
ys = map (+) xs
```

   (a) Vilken typsigantur har `ys`? *(2p)*

   (b) Vad returneras vid anropet `(head ys)`? *(3p)*

3. Följande kod representerar vektorer i planet och addition av två vektorer.

```
data Vector2 = Vector2 Double Double
             deriving (Eq, Show)

add (Vector2 x1 y1) (Vector2 x2 y2) =
  (Vector2 (x1 + x2) (y1 + y2))
```

   Följande kod läggs till (nedan kallad för "tilläggskoden"):

```
instance Num Vector2 where
  (+) = add
```

   (a) Vad blir konsekvensen av tilläggskoden? *(2p)*

   (b) Ge exempel på ett anrop där tilläggskoden kommer till användning. *(1p)*

Del 2: Logikprogrammering

4. **Kontrollflöde** *(8p)*

   Predikatet `member(X, L)` är sant om och endast om elementet `X` finns med i listan `L`. Predikatet kan definieras så här:

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

   Utifrån denna definition, beskriv kontrollflödet, unifieringarna samt resultatet vid frågan:

   ```
   ?- member(2, L), member(L, [[1, 2]]).
   ```

5. **Negation** *(2p)*

   Vad menas med antagandet om en sluten värld (eng. *closed world assumption*)? I detta sammanhang, förklara hur negation fungerar i Prolog, gärna med exempel.

6. **Rekursion, Backtracking**

   Vi vill skapa en kunskapsbas om städer i Sverige och hur man kan resa från en stad till en annan. Kunskapsbasen ska bestå av fakta som `edge(stockholm, uppsala)` som beskriver att det finns en direkt väg från Stockholm till Uppsala som inte passerar andra städer. Vi vill definiera ett predikat `route(+From, +To, ?Route)` som är sant om och endast om `Route` är en lista över städer som börjar med `From` och leder till `To` genom att följa direkta vägar och utan upprepning av städer.

   Betrakta följande implementation av `route(+From, +To, ?Route)`:

   ```
   route(From, To, [From, To]) :- edge(From, To).
   route(From, To, [From, Next | Route]) :-
      edge(From, Next),
      route(Next, To, [Next | Route]),
      \+ member(From, Route).
   ```

   (a) Tyvärr verkar inte denna implementation riktigt fungera som man tänkt sig när man försöker generera möjliga färdvägar från en stad till en annan. Förklara varför! Illustrera gärna ditt argument med ett konkret exempel (en samling fakta och en fråga). *(3p)*

   (b) Föreslå en ny implementation av predikatet `route`, som fungerar som önskat. *(7p)*

Del 3: Formella språk och syntaxanalys

7. **Reguljära språk**

   I denna uppgift arbetar vi med vägbeskrivningar i labyrinter (som t.ex. den som visas nedan). En beskrivning består av tecknen '`L`' (vänster), '`R`' (höger), '`U`' (upp) och '`D`' (ner). T.ex. tolkas "`LLDDDR`" som "gå två steg åt vänster, sedan tre steg nedåt, sedan ett steg åt höger".

   (a) Vi vill inte tillåta vägbeskrivningar där man går tillbaka samma väg som man kom ifrån. Med andra ord är t.ex. "`DLLUDR`" ej tillåtet eftersom vi i näst sista steget går nedåt direkt efter att ha gått uppåt. Däremot är det tillåtet att gå runt i en cirkel, t.ex. är "`LDRU`" en tillåten beskrivning. Är språket med vägbeskrivningar enligt denna begränsning reguljärt? Motivera ditt svar! *(3p)*

   (b) Nu vill vi inte ens tillåta att man går runt i cirklar – vägbeskrivningen får aldrig ta oss tillbaka till en position vi redan besökt. Vi har dock inga begränsningar på hur långt vi får gå, t.ex. är det tillåtet att ta en miljard steg åt vänster. Är språket med vägbeskrivningar enligt denna begränsning reguljärt? Motivera ditt svar! *(3p)*

   (c) Anta att vi specifikt arbetar med vägbeskrivningar i följande fixerade labyrint, där vi *startar i övre vänstra hörnet* ('`.`' representerar rutor man kan besöka, '`#`' representerar väggar).

```
###############################################
#......###......####.....####.....###......##
#.####..##.####..##..###..##..###..##.####..#
#.#####.##.#####.##.#####.##.#####.##.#####.#
#.#####.##.#####.##.#####.##.########.#####.#
#.####..##.####..##.#####.##.########.####..#
#......###......###.#####.##.########......##
#.########...######.#####.##.###...##.#######
#.########.#..#####.#####.##.#####.##.#######
#.########.##..####.#####.##.#####.##.#######
#.########.###..###..###..##..###..##.###...#
#.########.####..###.....####.....###.###.#.#
#.########.#####.#####.########.#####.###...#
#.....................................#######
###############################################
```

   En vägbeskrivning är bara giltig om den inte innehåller någon instruktion som skulle göra att vi går in i väggen. T.ex. är "`RRRRRDRDD`" och "`DDDDDDDDDDDDD`" giltiga, men "`U`" och "`RRRRDRDD`" är inte giltiga. Vi tillämpar inte reglerna från (a) och (b), det är nu tillåtet att gå tillbaka samma väg man kom. Är språket med giltiga vägbeskrivningar i labyrinten ovan reguljärt? Motivera ditt svar! *(3p)*

   (d) Vi kombinerar nu uppgift (b) och (c), och är ute efter vägbeskrivningar i den specifika labyrinten ovan (som i uppgift (c)), där det inte är tillåtet att besöka samma position mer än en gång (som i uppgift (b)). Är språket reguljärt? Motivera ditt svar! *(3p)*

8. **Grammatiker**

   Betrakta följande grammatik med slutsymbolerna `b`, `c`, `d`, `f`, `p`, och `z` (startsymbol är Start):

$$\begin{aligned} \mathsf{Start} &\rightarrow \mathtt{c}\ \mathsf{Cat} \mid \mathtt{d}\ \mathsf{Dog} \\ \mathsf{Cat} &\rightarrow \mathtt{p} \mid \mathtt{z} \mid \mathtt{f}\ \mathsf{Dog} \mid \mathsf{Start}\ \mathsf{Start} \\ \mathsf{Dog} &\rightarrow \mathtt{p} \mid \mathtt{z} \mid \mathtt{b}\ \mathsf{Cat} \mid \mathsf{Start} \end{aligned}$$

   (a) Ge tre exempel på strängar som tillhör språket, och ett exempel på en sträng som inte tillhör språket. *(1p)*

   (b) Förklara med pseudo-kod hur en rekursiv medåknings-parser för grammatiken skulle se ut. *(5p)*

   (c) Är grammatiken tvetydig? Motivera ditt svar! *(2p)*