Tenta i Programmeringsparadigm 2014-01-18 10.00-13.00

Inga hjälpmedel är tillåtna. Skriv inga svar på blanketten. Bonuspoäng från C/Inet-labbarna hösten 2013 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.**

---

1 Prolog

1.1 Lådmodellen (6 p)

Betrakta Prologprogrammet nedan:

```
isCitizen(tom).
hasPass(anna).
hasPass(pelle).
hasVisa(pelle).
canEnter(X) :- hasPass(X), !, hasVisa(X).
canEnter(X) :- isCitizen(X).
```

Rita lådmodellen och visa kontrollflödet vid evalueringen av `canEnter(X)` med hjälp av numrerade pilar (för varje steg av beräkningen) och respektive variabelunifiering.

1.2 Backtracking (4p)

Betrakta Prologprogrammet nedan:

```
predA([], Y, Y).
predA([H|T], Y, Z) :- predA(T, [H|Y], Z).
predB(X, Y) :- predA(X, [], Y).
```

Vilka blir det första och andra resultatet på målet `predB([5,3,7], L)`?

1.3 Negation (3p)

Varför måste man vara försiktig med negation i Prolog, om man t.ex. använder sig av regler av typen:

```
ogillar(X, Y) :- \+ gillar(X,Y).
```

1.4 Rekursion, Listor (7p)

Betrakta listor över positiva heltal.

(a) Skriv ett predikat `repl(X, N, NX)` som är sant när `NX` är listan som innehåller elementet `X` exakt `N` gånger och inga andra symboler. *(4p)*

(b) Med hjälp av predikatet ovan, skriv ett predikat `replicate(Xs, Ys)` som är sant när `Ys` är listan `Xs` där varje element `X` i `Xs` har replikerats exakt `X` gånger. *(3p)*

---

2 Haskell

2.1 (5p)

(a) Skriv en funktion `sameEl` i Haskell som kollar om samtliga tal i en godtycklig lista är lika eller ej. Ange även funktionens typsignatur! *(3p)*

Exempel:
`sameEl []` ger `True`
`sameEl [1]` ger `True`
`sameEl [1,2]` ger `False`
`sameEl [1.2, 1.2, 1.2]` ger `True`

(b) Hur ska funktionen i a)-uppgiften skrivas om för att vara så generell som möjligt dvs så att den också kan avgöra om tecknen i en godtycklig sträng är lika eller ej. Ange även funktionens typsignatur! *(2p)*

Exempel:
`sameEl ''hej''` ger `False`
`sameEl ''hh''` ger `True`

Obs! Exemplen i (a)-uppgiften gäller här också.

2.2 (10p)

I denna uppgift ska du skriva olika funktioner som byter ordningen inom ett talpar för alla talpar i en lista OCH filtrerar bort talpar som innehåller någon negativ komponent.

Exempel:
`changeOrd []` ger `[]`
`changeOrd [(1,2)]` ger `[(2,1)]`
`changeOrd [(3,0),(4,4),(-2,3),(34,23)]` ger `[(0,3),(4,4),(23,34)]`

(a) Skriv en rekursiv funktion `changeOrd` som löser problemet och INTE använder sig av den svansrekursiva idén, högre ordningens funktioner eller listomfattning. *(3p)*

(b) Visa vilka anrop som görs och i vilken ordning då följande beräkning görs (enligt a)-uppgiften): `changeOrd [(3,2),(-2,3),(34,23)]` ger `[(2,3),(23,34)]` *(1p)*

(c) Antag att vi redan har en funktion `nNeq` som avgör om komponenterna i ett par är icke-negativa, till exempel:
`nNeg (1,2)` ger `True`
`nNeg (1,-2)` ger `False`
`nNeg (-1,-2)` ger `False`
`nNeg (0, 2)` ger `True`
Skriv en funktion som löser problemet med hjälp av `map`, `filter`, `nNeg` och en anonym funktion som utför bytet av ordningen i talparet. *(3p)*

(d) Skriv en funktion som löser problemet med hjälp av listomfattning. *(3p)*

2.3 (5p)

Givet följande kod:

```
data Shape = Rectangle Float Float
           | Circle Float
           | Triangle Float Float
```

(a) Skriv en funktion `area` som räknar ut arean för de tre formerna `Rectangle`, `Circle` respektive `Triangle` (rätvinklig) där du använder dig av ovanstående givna kod. Anta att π finns definierad och heter `pi`. *(3p)*

(Vi erinrar oss att rektangelns area är produkten av sidlängderna, arean för en cirkel är π multiplicerat med radien i kvadrat och att den rätvinkliga triangelns area är bas multiplicerat med höjd delat med 2.)

(b) Ange hur man anropar funktionen `area` i (a)-uppgiften) för att beräkna arean för en triangel med basen 20 och höjden 10 respektive för att beräkna arean av en cirkel med radien 10. *(2p)*

---

3 Syntax

3.1 (4 p)

(a) Vad är ett reguljärt språk? *(1p)*

(b) Betrakta de språk som kan beskrivas med en kontextfri grammatik, samt de språk som kan beskrivas med en NPDA (*Non-deterministic Push-Down Automaton*). Vilken av dessa två språkklasser är störst (eller är de lika stora)? *(1p)*

(c) Varför är det ibland bra att göra *lexikal analys*? *(1p)*

(d) Vad är en härledning (eng. *derivation*) inom syntaxanalysen? *(1p)*

3.2 (8 p) Betrakta följande grammatik med startsymbol S, icke-slutsymboler S, T och U samt slutsymboler 'x', '(', ')' och ',':

$$S \to (\ T\ )$$
$$T \to U \mid U\ ,\ T$$
$$U \to x \mid S$$

(a) Beskriv vilket språk som grammatiken definierar. *(1p)*

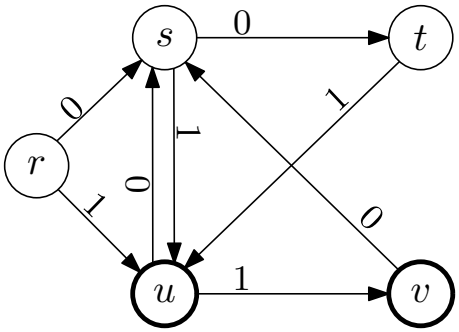
(b) Är grammatiken LL(1)? Motivera ditt svar. *(2p)*

(c) Är språket LL(1)? Motivera ditt svar. *(2p)*

(d) Välj ett indata på minst 9 slut-symboler som ligger i språket, och rita ett syntaxträd för detta. *(3p)*

3.3 (8 p)

Betrakta följande DFA med alfabete $\{0, 1\}$, starttillstånd r och där de accepterande tillstånden är u och v.



(a) Beskriv vilket språk automaten känner igen. *(2p)*

(b) Skriv ett reguljärt uttryck som beskriver samma språk, eller argumentera för att ett sådant inte finns. Du får använda syntaktiskt socker som "`?`" och "`+`". Du kan också använda RegEx-notation som "`[...]`", men du får inte använda bakåtreferenser ("`\1`", osv). *(5p)*

(c) Beskriv i ord vilket språk automaten skulle känna igen om det istället var t och v som var de accepterande tillstånden. *(1p)*
