Tentamen i DD1361 Programmeringsparadigm, 15 dec 2009

Kursansvarig: Lars Arvestad

- Detta är en **kamraträttad tenta**! Skrivtiden är 9.00 till 11.00 och rättningstiden är 11.15 till 12.00.
- Du ska ha **två tentaomslag**: ett anonymt och ett slutgiltigt. På båda omslagen och alla papper ska du en kod som består av sal och platsnummer. **Om du sitter på plats 17 i sal D31 ska du märka dina papper med "D31:17".**
- Det anonyma omslaget ska bara ha din kod, inte ditt namn.
- Det slutgiltiga omslaget ska ha ditt namn och din kod. Detta används när tentan lämnas till studentexpeditionen.
- **Skriv inte namn** på skrivpappren. Tentorna är anonyma! Istället skriver du sal och platsnummer.
- Lämna in tentan i det anonyma omslaget när du är klar och lämna sedan salen. Vi samlas sedan kl 11.00 i sal E1 för gemensam rättning! Varje tentand rättar då en annan tentands tenta med hjälp av rättningsmall. Rättningen kommer att granskas av Lars.
- För varje baslabb du har redovisat ges 1 bonuspoäng. För godkänt på tentamen krävs 20 poäng. Möjlighet till komplettering garanteras vid 19 poäng. Observera att tentamen inte betygsätts.
- **Ni som gick kursen innan avsnittet om syntaxanalys tillkom behöver inte göra sista avsnittet på tentan.** Er tenta omfattar alltså 32 poäng och för godkänt krävs 16 poäng. Komplettering garanteras vid 15 poäng.
- För alla tentander krävs det minst tre poäng per avsnitt på tentan.
- **Godkända hjälpmedel:** Skrivmedel. "Prolog programming" av Brna. "Programming in Haskell" av Hutton, eller det tidigare särtrycket av samma bok. "Functional Programming" av Jeroen Fokker och "An Introduction to Computing with Haskell" av Manuel Chakravarty går också bra.

Lycka till!

Allmänna frågor

1. Vad innebär "typhärledning"? Ge ett exempel, gärna med pseudokod. (2p)
2. Lat evaluering kan vara svårt att vänja sig vid, men förutom det inte uppskattas av alla finns det ibland nackdelar med lat evaluering. Nämn en! (2p)
3. (a) Vad innebär "semantik" när man diskuterar programmeringsspråk? (1p)
   (b) Vad innebär att ett typsystem är säkert? (1p)
4. Många moderna högnivåspråk har ett så kallat *foreign function interface* (FFI) som beskriver hur man gör anrop till funktioner skrivna i ett annat programmeringsspråk, eller definierar funktioner som kan anropas från ett annat programmeringsspråk. Med "annat programmeringsspråk" avses oftast C!

   Nämn två svårigheter som man måste ta hand om när man länkar ihop C-kod med kod i ex.v. Haskell, Lisp, Perl, Java, m.fl.! (2p)

Funktionell programmering i Haskell

5. (a) Antag du har funktionen `on :: Num a => a -> Bool`, vad är då typen för `filter on`? (1p)
   (b) Antag

```
annotate x = if x > 0::Int then
    (x, True)
  else
    (x, False)
```

   Vad är typen för `annotate`? (1p)

6. Minns du Newton-Raphsons metod för att hitta rötter till funktioner? Vi har en funktion $f(x)$ och dess derivata $f'(x)$, och vill hitta en lösning till ekvationen $f(x) = 0$. Vi har en hygglig gissning x_0 till lösningen. Med Newton-Raphson uppdaterar man sin gissning x_i genom att beräkna nästa värde $x_{i+1} = x_i - f(x_i)/f'(x_i)$. Om $|x_{i+1} - x_i| < \epsilon$, för nån önskad precision ϵ, svarar vi x_{i+1}, annars itererar vi med x_{i+1} som ny gissning.

   Skriv en enkel Newton-Raphson-lösare i Haskell! Din funktion `nr` ska arbeta med `Float`-värden och ta argumenten f, f', ett värde på precisionen, och ett x-värde.

   (a) Vad blir typsignaturen för `nr`? (1p)
   (b) Gör en enkel implementation av `nr`, d.v.s. ingen detektion av numeriska problem. (2p)

7. Skriv en funktion `boundingbox` som tar en lista L med heltalspunkter i planet och returnerar två punkter representerande en *bounding box* för L.

   Punkterna lagras som par med hjälp av

```
type Pkt = (Int, Int)
```

   och den resultaterande *bounding box* ska ge två motsatta hörn i en rektangel som inbegriper alla punkter i L. (3p)

   Exempelkörning:

```
Main> boundingbox [(0,0)]
((0,0),(0,0))
Main> boundingbox [(1,1), (2,2)]
((1,1),(2,2))
Main> boundingbox [(1, 0), (2, 1), (1, 2), (0, 1), (1, 1)]
((0,0),(2,2))
```

8. Skriv en funktion `search` som letar upp element i en lista och håller reda på hur ofta olika sökningar har gjorts. Listan lagrar element av den parameteriserade typen `[Elem a]` enligt följande definition:

```
data Elem a = Elem a Int
```

   Det första fältet i `Elem` är ett värde och det andra en räknare.

   Funktionen `search` ska returnera resultatet av en sökning samt en version av indatalistan där räknarna har uppdaterats. Typsignaturen är

```
search :: Eq a => a -> [Elem a] -> (Maybe Int, [Elem a])
```

   Exempelkörning: (4p)

```
Tenta> search 17 []
(Nothing,[Elem 17 1])
Tenta> search 17 [Elem 17 1]
(Just 1,[Elem 17 2])
Tenta> search "KTH" [Elem "LTH" 2, Elem "CTH" 1]
(Nothing,[Elem "LTH" 2,Elem "CTH" 1,Elem "KTH" 1])
```

Logikprogrammering med Prolog

9. Skriv ett predikat `positivt` som avgör (lyckas) om ett tal är större än noll eller och misslyckas annars. (1p)

10. Låt heltalsmängder representeras av listor av heltal. Implementera ett predikat `union/3` som returnernar unionen av två mängder i den tredje parametern. (3p)
Exempelkörning:

```
| ?- union([1,2], [3,4], S).
S = [1,2,3,4] ? n
no
| ?- union([1,2], [], S).
S = [1,2] ? n
no
| ?- union([1,2], [2,3], S).
S = [1,2,3] ? n
no
```

11. (a) Skriv ett predikat `genereramera` som tar två parametrar: en lista som indata och en utdataparameter. Vid upprepat anrop ska predikatet returnera varje element i indatalistan. (1p)
Exempelkörning:

```
| ?- genereramera([1, 2, 3], X).
X = 1 ? n
X = 2 ? n
X = 3 ? n
no
```

(b) Modifiera din lösning så att den inte genererar *något* svar om listan innehåller ett negativt värde, men fungerar som tidigare om det bara är icke-negativa värden i listan. (2p)
Exempelkörning:

```
| ?- genereramera([1, 2, 3], X).
X = 1 ? n
X = 2 ? n
X = 3 ? n
no
| ?- genereramera([1, 2, -3], X).
no
```

12. (a) Skriv ett predikat som beräknar kvadraten på skillnaden mellan två tal. (1p)

(b) Skriv ett predikat `rmsd` som tar tre parametrar: två listor av tal och en returparameter som är roten av summan av skillnaderna mellan elementen i de två listorna. Du kan använda att det finns en funktion `sqrt` som är tillämpbar i *aritmetiska uttryck*. (4p)
Exempel: RMSD på två listor $[1, 0, 1, 0]$ och $[0, 1, 0, 1]$ är 2. RMSD på två identiska listor är 0.

Syntaxanalys

13. Grammatiker brukar definieras rekursivt med hjälp av "regler". Man använder en speciell terminologi när grammatikerna och deras regler diskuteras.

    (a) Vad kallas normalt dessa "regler"? (1p)

    (b) Vad kallas de atomära delarna, "symbolerna", i reglerna? (1p)

14. Definiera reguljära uttryck som till fullo matchar de givna språken.

    (a) (1p)
    ```
    xyx
    xyyyx
    xyxyx
    xyyxxyyx
    ```

    (b) I arvsmassan på större flercelliga organismer är så kallade tandemupprepningar vanliga. Dessa upprepningar består av många korta kopior av en liten sträng som ligger bredvid varandra.

    Vi är nu intresserade av alla upprepningar av två nukleotider (representerade av bokstäverna A, C, G, och T), och *inga* andra mönster. (1p)

15. Det reguljära uttrycket

    ```
    ([xy][yz])|(x[wyz])
    ```

    genererar ett språk. Hur många ord finns det i det? (1p)

16. *Symbolic expressions*, även kända som S-uttryck eller *sexp*, har länge använts för att lagra strukturerade data. De är kända framförallt ifrån LISP och dess dialekter, där S-uttryck har använts för att lagra både data och kod. De kan vara mycket bekväma att använda eftersom stora datastrukturer kan, likt XML-baserade data, både skrivas och läsas med enkla instruktioner som 'read' och 'write'. Listor i LISP lagras som S-uttryck.

    Strukturen på S-uttryck är enkel. En atom, d.v.s. ett tal, en sträng, eller en symbol, är alltid ett S-uttryck. Dessutom är tomma listan, skriven som `()`, en atom.

    Man kan sätta ihop två S-uttryck s_1 och s_2 till ett nytt S-uttryck med två parenteser och en punkt: $(s_1\ .\ s_2)$. Det finns syntaktiskt socker som låter $(s_1\ .\ (s_2\ .\ (s_3\ .\ ())))$ ersättas av $(s_1\ s_2\ s_3)$. Notera att det då inte finns några punkter eller kommatecken, och att `()` används för att indikera "slut på listan".

    Din uppgift är att skriva en kontextfri grammatik på Extended Backus-Naur-form som beskriver S-uttryck. De ska kunna innehålla heltal, strängar (på formen ``` ``hej'' ```), och symboler (utan citattecken: `enSymbol`). Du behöver inte beskriva hur dessa atomer är uppbyggda, men behöver då tydligt ange vilka antaganden du gör, dvs hur du anger atomerna i din grammatik. (3p)