Tentamen i 2D1361 Programmeringsparadigm, 21 november 2006

Kursansvarig: Lars Arvestad

Skriv tydligt! Skriv bara på en sida av pappret och behandla bara en uppgift per pappersblad. Ge dina svar tydliga motiveringar. Förklara dina funktioner och predikat! Lämna plats för kommentarer vid rättning.

För godkänt krävs 30 poäng, 40 poäng ger betyg 4, och vid 50 poäng ges betyg 5. För varje labb redovisad i tid ges *två* bonuspoäng. (Jag ökade tentans poängomfång från 30 till 60 poäng, och dubblar därför även bonuspoängens värde.)

Ni som gick kursen innan avsnittet om syntaxanalys stoppades in behöver inte göra sista avsnittet på tentan. Er tenta är alltså på 52 poäng. För er är gränserna godkänt vid 26 poäng, 35 poäng ger betyg 4, och vid 44 poäng ges betyg 5.

Lösningsförslag kommer att hittas på kursens hemsida.

Godkända hjälpmedel: Skrivmedel. "Prolog programming" av Brna. "Programming in Haskell" av Hutton. "The Haskell School of expression" av Hudak. "Lite om syntaxanalys" av Kusoffsky.

Lycka till!

Allmänna frågor

Samla gärna svaren på detta avsnitt på ett papper.

1. (a) Vad innebär syntaktiskt socker? Förklara och ge ett exempel. (2p)
   (b) Vad är ett idiom? Förklara och ge ett exempel. (2p)
2. (a) Förklara begreppet statisk typning. (2p)
   (b) Vad menas med att ett programmeringsspråk är *starkt* eller *svagt* typat? Jämför typsystemen i C och Haskell i din förklaring. (2p)
3. Vad innebär "lat evaluering"? (1p)
4. Vad innebär begreppet "imperativt programmeringsspråk"? (1p)

Funktionell programmering i Haskell

5. Beskriv uttrycket `f = map length`. Vad är resultatet och hur kan man använda `f`? (2p)
6. (a) Skriv en funktion `tentakul :: Eq a => [a] -> [a] -> [(a,a)]` som tar två listor och returnerar en lista med alla par där första elementet är från första listan och andra elementet är från den andra listan, med förbehållet att parens två elementen är olika. Du ska lösa uppgiften med listomfattning (*list comprehensions*). (3p)

   Exempelkörning:

   ```
   > tentakul [1,2] [1,3]
   [(1,3),(2,1),(2,3)]
   > tentakul [1,2] [1,2]
   [(1,2),(2,1)]
   > tentakul ['a', 'b', 'c'] ['a']
   [('b','a'),('c','a')]
   ```

   (b) Förklara vad "`Eq a =>` betyder i funktionens typsignatur. (1p)

7. Den här uppgiften rör en sport vi kallar *generaliserad bowling* där man spelar en serie om k kast. Varje kast i ger ett *resultat* $x_i \in \{0, 1, \ldots, 10\}$ och en *poäng* p_i som beräknas så här:

   (a) *Strajk:* Om $i < k$ och $x_i = 10$ är $p_i = \sum_{j=i}^{\min(i+2,k)} x_j$, dvs påföljande två kast räknas dubbelt.

   (b) *Strajk sist:* Om $i = k$ och $x_i = 10$ är $p_i = 20$.

   (c) Annars är $p_i = x_i$.

   Totalpoängen för en serie är $\sum_{i=1}^{k} p_i$. Du ska skriva en funktion i Haskell med signaturen

   ```
   bowling :: [Int] -> Int
   ```

   som beräknartotalpoängen för en lista med korrekta resultat. (5p)

```
> bowling [10]
20
> bowling [10, 1]
12
> bowling [10, 1, 1]
14
> bowling [10, 1, 1, 1]
15
> bowling [1, 10, 1, 1]
15
> bowling [1, 1, 10, 1]
14
> bowling [10, 10, 10, 10]
100
```

Exempelkörningar för uppgift 7.

8. (a) Definiera en datatyp `Polynomial` som lagrar ett polynom som en lista av dess koefficienter, med lägst grad först. Dessutom anges polynomets grad i datatypen så att vi slipper att beräkna graden från listan varje gång det efterfrågas. Graden anges med ett heltal och koefficienterna ska lagras som flyttal. (2p)

   (b) Skriv en funktion `coefficient :: Polynomial -> Int -> Float` som returnerar en koefficient i ett polynom. Om variabeln `poly` representerar polynomet $1 + 2x + 3x^2 + 4x^3$ ska funktionen fungera så här:

   ```
   > coefficient poly 0
   1
   > coefficient poly 3
   4
   ```

   (2p)

   (c) Skriv en funktion `evaluate` som beräknar ett polynoms värde i en given punkt. Typsignaturen är alltså `evaluate :: Polynomial -> Float -> Float`. Exempelanvändning:

   ```
   > evaluate poly 0
   1
   > evaluate poly 1
   10
   ```

   (3p)

9. Pseudoslumptalsgeneratorer (PSG) används för att generera sekvenser av tal som ska uppträda som om de vore utfall från en stokastisk variabel. En enkel sådan PSG är den linjära kongruensgeneratorn

$$r_i = Ar_{i-1} + B \pmod{M}$$

   (som numera anses vara ganska dålig), för $i > 0$, och startvärdet r_0 (även kallat *fröet*) som användaren föreslår. A, B och M är alla parametrar hos generatorn.

   (a) Skriv en funktion `psg` som tar r_0, A, B och M som argument och returnerar en ström av pseudoslumptal från en linjär kongruensgenerator. Ange även funktionens (`psg`) typsignatur. Användingsexempel: (3p)

   ```
   > take 10 (psg  3 5 7 9)
   [4,0,7,6,1,3,4,0,7,6]
   ```

   (b) Skriv en högre ordningens funktion `metapsg` som tar en rekursion som första argument och ett frö som andra argument och returnerar en ström av pseudoslumptal. (2p)

   Användingsexempel:

   ```
   > take 10 (metapsg rekursion 3)
   [4,0,7,6,1,3,4,0,7,6]
   ```

   där vi redan har definierat

   ```
   rekursion r = (5 * r + 7) `mod` 9
   ```

Logikprogrammering med Prolog

10. Vad är skillnaden mellan *logisk läsning* och *procedurell läsning*? (2p)

11. Betrakta följande Prologpredikat.

```
% Remove elements equal to the second parameter
% from the list in the first parameter, result
% is placed in the third parameter.
remove([], _, []).
remove([E|Xs], E, Ys) :- !, remove(Xs, E, Ys).
remove([X|Xs], E, [X|Ys]) :- remove(Xs, E, Ys).
```

    (a) Sats 2 innehåller ett *snitt* (eng: *cut*). Klassas det som "rött" eller "grönt" snitt? Motivera ditt svar. (2p)

    (b) Vad är resultatet av anropet

    ```
    remove(L, 1, [2,3,4]).
    ```

    där L är ounifierad? (2p)

12. Antag att vi har koden

    ```
    last([X], X).
    last([_|Xs], X) :- last(Xs, X).
    ```

    Illustrera kontrollflödet för exekveringen

    ```
    | ?- last([1, 2], X).
    X = 2 ?
    yes
    ```

    med hjälp av lådmodellen! (4p)

13. Pascals triangel är välkänd, kanske mest tack vare dess koppling till binomialkoefficienterna. Ett element i triangeln är summan av de två elementen ovanför, om de existerar, annars är elementet 1.

    I den här uppgiften ska du skriva ett predikat `pascal/2` vars första parameter är ett heltal som talar om vilken *rad* i Pascals triangel som ska returneras i den andra parametern. (5p)

```
                1
             1     1
          1     2     1
       1     3     3     1
    1     4     6     4     1
```

De fem första raderna i Pascals triangel.

Exempelkörning:

```
| ?- pascal(1, L).
L = [1] ? n
no
| ?- pascal(2, L).
L = [1,1] ? n
no
| ?- pascal(5, L).
L = [1,4,6,4,1] ?
```

14. För att implementera en kö i Prolog använder man lämpligen differenslistor. För att skapa sin kö behöver man ett predikat `makeQueue`/1 som unifierar sitt argument till en tom kö, dvs en tom differenslista. Ni kan använda definitionen `makeQueue(X-X).`

    Din uppgift är att implementera predikaten `enqueue`/3 och `removeFirst`/3:

    `enqueue`/3 tar en kö Q som första argument, ett element X som andra argument, och unifierar tredje argumentet till en kö med X instoppat *sist* i Q.

    `removeFirst`/3 tar en kö Q som första argument och unifierar andra argumentet med första elementet i Q. Dess tredje argumentet unifieras till kön Q med första elementet borttaget.

    Exempel:

```
| ?- makeQueue(Q1), enqueue(Q1, lasse, Q2), enqueue(Q2, berra, Q3),
     removeFirst(Q3, X, Q4), removeFirst(Q4, Y, Q5).
X = lasse
Y = berra
Q1 = ...
...
Q5 = _A-_A ?
```

    (4p)

Syntaxanalys

Du som följde kursen innan 2005 struntar i detta avsnitt!

15. (a) Skriv ett reguljärt uttryck som känner igen språket av namn som rimmar på "ika". Om det finns en bokstav före "ika" måste det vara en konsonant. Namn börjar alltid på stor bokstav! Vi är i övrigt mycket generösa med bedömningen av vad som är ett namn. Några exempel på namn som din lösning ska känna igen: Monika, Monica, Ika, Annica, Mika. Några som *inte* ska matcha: Moniqa, Lasse, Laika. (2p)

    (b) Skriv ett reguljärt uttryck som känner igen alla felstavningar av "Lasse" sådana att *en* bokstav har råkat bli ersatt av fel bokstav, eller att "ss" har blivit "s". Dessa strängar ska tillhöra språket: lasse, Lase, Lesse, Hasse, Lasso. Däremot är Lisa, Laasse, Laase och Lessa *inte* felstavningar som tillhör språket eftersom de innehåller minst två fel och/eller felstavningstyper vi inte bryr oss om. (2p)

16. Skriv en grammatik på BNF-form som beskriver den personallista i förenklat XML-format som visas här intill.

    Beskriv vilka terminaler du använder. Din grammatik ska beskriva en lista med obegränsat antal personer, och kunna hantera "tomma listan" utan några personer.

    Du ska alltså inte skriva ett prologprogram, utan en enkel grammatik som beskriver syntaxen på en datafil. (4p)

```
<staff>
  <person position=administrator>
    <firstname>Ada</firstname>
    <lastname>Slowmove</lastname>
    <ssid>720130-0123</ssid>
  </person>
  <person position=manager>
    <firstname>Betty</firstname>
    <lastname>Boss</lastname>
    <ssid>600123-4567</ssid>
  </person>
</staff>
```

Datafil för en personallista.