SI-möte #4, Programkonstruktion och datastrukturer

Elias Castegren & Kristiina Ausmees
elca7381@student.uu.se ‖ krau6498@student.uu.se

Begrepp

i) Iteration (svansrekursion) ii) Ackumulator iii) `datatype`-deklarationer
iv) Uppräkningsdatatyper v) Rekursiva datatyper vi) `option`-datatypen

Övningar

1.

i) Skriv en icke-svansrekursiv funktion `rovar : string->string` som översätter en sträng till rövarspråket. I rövarspråket dubblerar man varje konsonant och lägger in ett "o" mellan dubbletterna. Vokalerna lämnas oförändrade. T.ex. blir "program" på rövarspråket "poprorogogroramom". Du får använda funktionen `isVowel : char->bool` som returnerar true om `c` är en vokal. Skriv funktionsspecifikationer innan du skriver koden.

ii) Skriv en svansrekursiv version av samma funktion. Vilka ändringar behöver man göra i funktionsspecifikationen?

iii) Anropa båda funktionerna med strängen "hej". Hur ser det delvis beräknade uttrycket ut när man når basfallet i de olika versionerna?

2.
Betrakta nedanstående datatyper:

```
datatype flavour = Sweet | Sour | Bitter;

datatype fruitNVeg =
    Vegetable of string*real | Fruit of string*real*flavour;
```

Varje frukt och grönsak har ett namn och en vikt. Frukterna har dessutom en smak. Skriv intern dokumentation till datatypen `fruitNVeg`.

3.
Implementera följande funktioner (med funktionsspecifikationer) till datatypen `fruitNVeg`:

i) `getWeight(l):fruitNVeg list->real`, som beräknar den totala vikten av alla frukter och grönsaker i listan `l`

ii) `getVegetables(l):fruitNVeg list->fruitNVeg list`, som returnerar alla grönsaker i listan `l`

iii) `filterFlavour(l, f):fruitNVeg list*flavour->fruitNVeg list` som returnerar en ny lista med alla frukter från `l` som har smaken `f`.

4.
Utgå från nedanstående datatypsdeklaration

```
datatype 'a tree = Tree of 'a*('a tree) list;
```

och skriv en funktion `distance(v, t):'a*('a tree)->int option` som returnerar avståndet från roten i trädet `t` till noden med värdet `v` (alla värden i `t` kan antas vara unika). Avståndet ska vara taggat med `SOME`. Om det inte finns någon nod med värdet `v` skall `NONE` returneras.

5.
Skriv ett program (med komplett intern dokumentation) med lämpliga datatyper som kan avgöra utgången i en omgång sten-sax-påse. Reglerna i spelet är att sten slår sax, sax slår påse och påse slår sten. Två lika händer blir oavgjort.

6.
Skriv en svansrekursiv funktion `truthDistance(l):bool list->int list` som beräknar en lista med avstånden från varje förekomst av `true` till nästa likadana värde. Till exempel ska listan `[false, true, false, false, true, false, true, true]` beräknas till `[2,1,0]`.

***7.**
Eftersom alla element i en lista i ML måste vara av samma typ får man till exempel inte skriva ett uttryck som ser ut så här: `[1,2,[3,4,[1,2,3]],3]`. Skriv en datatyp `'a nested` för att representera sådana listor (där varje element är ett värde eller en lista av sådana värden). Notera att man fortfarande inte kommer kunna skriva nästlade listor med klamrar.

Skriv sedan en funktion `flatten(l):'a nested->'a list` som tar en nästlad lista och plattar till den, alltså "tar bort" alla listklamrar utom de yttersta. Ett anrop till `flatten` med den nästlade listan ovan ska ge resultatet `[1,2,3,4,1,2,3,3]`.

Lycka till!