

TENTAMEN I CDT 201 (f.d. CD5100) FUNKTIONELL PROGRAMMERING

Fredagen den 18 januari 2008, kl 8:30 – 13:30

Kurslitteratur är inte tillåten, och inte eller andra hjälpmedel som på något sätt kan ersätta kurslitteraturen (t.ex. egna anteckningar, andra böcker i ämnet, kopior av OH-bilder, datorer eller räknare med dito lagrad information). Endast generella hjälpmedel är tillåtna, som räknare utan lagrad information av betydelse för kursen, ordbok, allmän formelsamling och liknande. För godkänt krävs 15 poäng, max är 30 poäng. Resultatet offentliggörs senast fredagen den 8 februari 2008.

Vänligen observera följande:

- Motivera alltid dina svar. Bristande motivering kan ge poängavdrag. Omvänt kan även ett felaktigt svar ge poäng, om det framgår av motiveringen att tankegången ändå är riktig.
- Skriv tydligt!
- Varje blad skall vara försedd med namn, personnummer, uppgiftsnummer och bladnummer.
- Endast en uppgift på ett och samma blad.
- Skriv enbart på ena sidan av ett blad.
- Uppgifterna är inte nödvändigtvis sorterade i svårighetsgrad. Om du kör fast kan det löna sig att gå vidare till nästa uppgift.
- Lösningförslag kommer att finnas på kursens hemsida efter att tentan är slut.

Frågor: Björn Lisper på 021-151709.

UPPGIFT 1 (6 POÄNG)

Horners regel är en effektiv algoritm, där man räknar ut värdet av ett polynom $a_0 + a_1x + a_2x^2 + \dots + a_nx^n$, för ett givet x , som $a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + xa_n) \dots))$.

a) Skriv en funktion `horner x l` som tar ett värde `x` och en lista `l = [a0, a1, ..., an]` av koefficienter och räknar ut polynomets värde för x enligt Horners regel!. Använd direkt rekursion. (3p)

b) Gör en alternativ funktionsdefinition för Horners regel som istället använder någon eller några av Haskell's inbyggda högre ordningens funktioner! (3p)

UPPGIFT 2 (6 POÄNG)

a) Notationen `[1 . .]` är en kortform för någonting i Haskell. Vad? (1p)

b) Skriv en rekursiv definition i Haskell som räknar ut samma sak som `[1 . .]`. (3p)

c) Vad blir resultatet i Haskell av att räkna ut `take 5 [1 . .]`? Varför? Ordentlig motivering krävs för full poäng! (2p)

Ledning: du kan förutsätta följande (förenklade) deklARATION av `take`:

```
take 0 xs      = []
take n (x:xs) = 1 + take (n-1) xs
```

Om du inte vet vad `[1..]` står för kan du ersätta den med någon valfri oändlig lista i uppgift *c*).

UPPGIFT 3 (4 POÄNG)

Haskell har en funktion `sequence_ :: [IO a] -> IO ()` som utför en lista av actions i den ordning de ligger i listan, och "returnerar" `()`. Ge en definition i Haskell av `sequence_!`. Definiera sen Haskell's standardfunktion `putStr` med hjälp av `sequence_` och `putChar`.

UPPGIFT 4 (4 POÄNG)

Skriv en Haskellfunktion som tar en sträng innehållande en text och gör om texten så att varje ord som inleder en ny mening börjar med stor bokstav! Du får använda följande enkla regel för att avgöra början på en ny mening: det första ordet i en ny mening påbörjas av det första alfabetiska tecken som kommer efter ett sluttecken på en mening (`.`, `!`, `?`). Du får också anta att strängen inleds med en mening som börjar med ett ord, utan några andra inledande tecken.

Ledning: i standardbiblioteket (modulen) `Char` finns två funktioner som du kan använda:

- `isAlpha :: Char -> Bool`, testar om argumentet är ett alfabetiskt tecken eller ej, och
- `toUpper :: Char -> Char`, konverterar argumentet till stor bokstav.

Du måste dock göra vederbörliga deklARATIONER i din kod för att få utnyttja dessa funktioner.

UPPGIFT 5 (6 POÄNG)

- a) Deklarera en datatyp `IntTree` för binära träd som lagrar heltal av typ `Int!`. Heltalen ska lagras i löven. (1p)
- b) Deklarera en klass `Ints` för datastrukturer som innehåller heltal av typ `Int!`. Klassen ska ha en metod, `largest`, som returnerar det största heltalet i en sådan datastruktur. (2p)
- c) Gör din datatyp `IntTree` till instans av klassen `Ints`. (3p)

UPPGIFT 6 (4 POÄNG)

Bevisa, för alla ändliga listor `l`, att

```
map id l = l
```

Där `id` definieras som

```
id x = x
```

Lycka till!

Björn