

# TENTAMEN I CD5100 FUNKTIONELL PROGRAMMERING

Torsdagen den 17 augusti 2006, 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 torsdagen den 7 september 2006.

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.

Jour: Björn Lisper på 070-6729577.

---

---

## UPPGIFT 1 (6 POÄNG)

Som bekant kan heltal representeras i binär representation. En binär representation kan i sin tur ges av en lista av nollor och ettor. Antag att vi representerar icke-negativa heltal med sådana listor, där de mer signifikanta bitarna är lagrade först i listan. T.ex. representerar  $[1, 1, 0]$  heltalet  $6 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$ .

a) Definiera en funktion `bs2int`, som konverterar en lista av nollor och ettor enligt ovan till ett heltal! (3p)

b) Definiera en funktion `int2bs`, som omvänt konverterar ett heltal till en lista av nollor och ettor enligt ovan! (3p)

## UPPGIFT 2 (3 POÄNG)

Betrakta följande funktionsdeklarationer:

```
fst (x,y) = x
f 0 = 1
f x = 2 * f (x-1)
```

Vad blir resultatet av `fst (17, f (-3))` i Haskell? Motivera ditt svar, och beskriv hur evalueringen går till.

## UPPGIFT 3 (4 POÄNG)

Vektorer kan representeras av listor av tal. Varje komponent i vektorn representeras då av ett element i listan. T.ex. listan  $[1, 0, -1]$  representerar den tredimensionella vektorn  $(1, 0, -1)$ .

Två vektoroperationer är *addition* av två vektorer och *skalning* av en vektor med en skalär. Definiera två funktioner `vadd` och `scale` som implementerar vektoraddition och skalning på vektorer representerade som listor! Exempelvis ska gälla att `vadd [1,0,-1] [2,1,1] = [3,1,0]` och `scale 3 [1,0,-1] = [3,0,-3]`. Dina operationer ska (givetvis) fungera för vektorer av godtycklig dimension.

För full poäng ska din lösning använda Haskell's inbyggda högre ordningens funktioner på ett väsentligt sätt. Dessutom ska fallet att man försöker addera vektorer av olika dimension hanteras på ett vettigt sätt: du får själv bestämma hur, men måste motivera väl varför du valt att göra som du gjort.

#### UPPGIFT 4 (5 POÄNG)

Det finns många andra sätt att representera vektorer än listor. Det kan vara trevligt att programmera med vektorer utan att behöva välja representation i förväg. Ett sätt att åstadkomma detta är att använda Haskell's typklasser och göra funktionerna till klassmetoder. Definiera nu en typklass `Vector` för vektorer. Klassen ska ha en metod `vadd`, som i förra uppgiften, som ska implementera vektoraddition. Gör sedan listor av numeriska data till en instans av klassen `Vector`. Du kan återanvända din lösning från förra uppgiften om du vill, men uppgiften kan också lösas utan att ha löst den uppgiften.

#### UPPGIFT 5 (2 POÄNG)

Kan man i Haskell skriva

```
[getChar, getChar, getChar, getChar, getChar]
```

för att läsa in fem stycken tecken? Motivera ditt svar!

#### UPPGIFT 6 (7 POÄNG)

Ett "trinärt träd" är ett träd där varje intern nod har tre söner.

a) Deklarera en datatyp `TTree` i Haskell för trinära träd, där man kan lagra ett datum i varje nod (både interna noder och löv)! De lagrade datumen ska kunna vara av vilken typ som helst (men av samma typ för ett och samma träd). (2p)

b) Definiera en funktion `sumtree` som summerar innehållet i noderna i ett trinärt träd! (2p)

c) Generalisera `sumtree` till en högre ordningens funktion `foldtree`, som applicerar en godtycklig funktion med två argument på innehållet i noderna i ett trinärt träd! (3p)

#### UPPGIFT 7 (3 POÄNG)

I Haskell använder man oftast pattern-matching när man programmerar med listor, för att ta ut huvudet och svansen på listan. Ett alternativ är att använda funktioner `head` och `tail` som returnerar huvudet och svansen, respektive, av sina argument. `head` kan i Haskell definieras sålunda:

```
head (x:xs) = x
head [] = error "Cannot take head of empty list"
```

Med denna deklaration, vad har `head` för mest generella typ? Ordentlig motivering krävs för full poäng.

**Ledning:** `error` har typen `String -> a`, för godtycklig typ `a`.

Lycka till! Björn