TENTAMEN I DVA 201 FUNKTIONELL PROGRAMMERING MED F#

Torsdagen den 7 januari 2016, kl 8:10 – 12:30

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 28 januari 2016.

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 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.
- Lösningsförslag kommer att finnas på kursens hemsida efter att tentan är slut.

Frågor: Björn Lisper på 021-151709.

UPPGIFT 1 (4 POÄNG)

Deklarera en polymorf funktion `interleave` som tar två listor som argument och returnerar en lista där vartannat element tagits från den första argumentlistan och vartannat från den andra. Om listorna är olika långa ska resterande element läggas till på slutet. T.ex. ska gälla att

```
interleave [1;2;3;4] [5;6] = [1;5;2;6;3;4]
```

UPPGIFT 2 (2 POÄNG)

Betrakta följande F#-deklarationer:

```
let f x = x + x
let g x = x + (x : float)
```

Vilka typer kommer funktionerna `f` och `g` att få och varför?

UPPGIFT 3 (4 POÄNG)

Det finns en känd mängd talföljder, som man bildar på följande sätt:

- startvärdet är ett positivt heltal,
- om ett tal x i följden är lika med 1, så finns inga efterföljande tal i talföljden,
- om ett tal x i följden är udda ($\neq 1$), så är nästa tal $3x + 1$, och

- om ett tal x i följden är jämnt, så är nästa tal $x/2$.

Man tror att *oavsett vilket positivt heltal som talföljden startar med, så kommer den förr eller senare ner till talet ett*. Ingen har hittills hittat ett motexempel. Å andra sidan har ingen lyckats bevisa påståendet heller! Detta är alltså fortfarande ett öppet problem inom matematiken.

Deklarera en funktion i F# som, för ett givet positivt heltal, returnerar antalet element i den talföljd som har heltalet som startvärde!

UPPGIFT 4 (2 POÄNG)

Förklara vad lat evaluering och ivrig evaluering (lazy and eager evaluation) är för något, och hur de skiljer sig åt! Har F# lat eller ivrig evaluering?

UPPGIFT 5 (4 POÄNG)

Deklarera en funktion `add_opt_array` som tar en array av typ `float option []` som argument och returnerar summan av de element som har formen `Some x` (där `x` är ett flyttal). För full poäng ska du använda några av de inbyggda högre ordningens funktionerna på arrayer på ett vettigt sätt.

UPPGIFT 6 (4 POÄNG)

Deklarera en funktion `printlist` som tar en lista och som sidoeffekt skriver ut elementen i listan ett och ett, på ny rad, föregånget av positionen i listan samt ett kolon plus mellanslag. T.ex. ska anropet `printlist [13;3;34]` ge utskriften

```
1: 13
2: 3
3: 34
```

Din lösning ska använda rekursion.

UPPGIFT 7 (6 POÄNG)

a) Deklarera en polymorf datatyp för binära träd, där noder kan ha 0, 1 eller 2 söner och data lagras i alla noder. *(2p)*

b) Deklarera en funktion som tar ett träd av typen som deklarerats i *a)*, och som returnerar en lista med alla element i trädet i "inorder". (Dvs. för varje nod först elementet i noden, sen (om existerar) noderna i vänstra delträdet i inorder, sen (om existerar) noderna i högra delträdet i inorder.) Om en nod bara har en son kan den sonens träd ses som antingen vänster eller höger delträd, det gör ingen skillnad. *(4p)*

UPPGIFT 8 (4 POÄNG)

Bevisa att följande likhet gäller:

```
List.map id = id
```

där `id` är identitetsfunktionen för vilken gäller att

```
id x = x
```

för alla `x`. Full poäng kräver ett formellt bevis med induktion. Ett mer informellt resonemang kan också ge poäng, men inte full pott.

Observera att `id` är en polymorf funktion. I likheten kommer den att ha olika typ: om den i vänstra ledet har typen `'a -> 'a` så har den i högerledet typen `'a list -> 'a list`.

Lycka till! *Björn*