TENTAMEN I DVA 201 FUNKTIONELL PROGRAMMERING MED F#

Fredagen den 5 juni 2015, kl 14:10 – 18:30

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 26 juni 2015.

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 (7 POÄNG)

`List`-modulen innehåller en funktion `List.exists : ('a -> bool) -> 'a list -> bool` som tar ett predikat `p` och en lista `l` som argument och som returnerar `true` om predikatet blir sant för något element i listan och `false` annars. Till exempel gäller att `List.exists (fun x -> x = 5) [1;3;5;3]` = `true` och `List.exists (fun x -> x = 5) [1;3;2;3]` = `false`.

a) Ge en deklaration av `List.exists` som använder direkt rekursion. *(4p)*

b) Ge en deklaration som använder inbyggda högre ordningens funktioner på listor på ett icke-trivialt sätt. *(3p)*

UPPGIFT 2 (2 POÄNG)

Betrakta följande F#-deklarationer:

```
let f x = x + x
let g x = f (x + 2.0)
```

Vad händer om man försöker kompilera dessa som de står, och varför? Förklara!

UPPGIFT 3 (4 POÄNG)

Deklarera en funktion `optadd : (int option list) -> (int option list) -> int list` som adderar två listor av typen `(int option list)` elementvis, på följande sätt:

- om de matchande elementen i listorna har formen `Some x` och `Some y` så ska den resulterande listan ha elementet `x + y` på motsvarande plats,
- om det ena elementet har formen `Some x` och det andra är `None` så ska den resulterande listan ha elementet `x` på motsvarande plats, samt
- om båda elementen är `None` så ska den resulterande listan ha elementet `0` på motsvarande plats.

Man behandlar alltså `Some x` som `x` och `None` som `0`. Till exempel ska gälla att `optadd [Some 3; None; None] [Some 5; Some 1; None] = [8; 1; 0]`. Du får anta att listorna är lika långa.

UPPGIFT 4 (3 POÄNG)

Betrakta följande F#-deklarationer:

```
let rec f x = 1 + f (x-1)
let g y = if y = 0 then 1 else f 1
```

a) Vad blir resultatet av anropet `g 0` om vi använder ivrig evaluering (call-by-value)? *(1p)*

b) Vad blir resultatet av anropet `g 0` om vi använder lat evaluering (call-by-need)? *(1p)*

c) Använder F# lat eller ivrig evaluering? *(1p)*

UPPGIFT 5 (4 POÄNG)

Deklarera en funktion `writearray name a`, där `name` är en sträng och `a` är en array av heltal (`int`). Funktionen ska skapa en fil med namnet `name` och skriva ut heltalen i `a` till filen som strängar, ett heltal per rad, i den ordning de ligger i `a`. Du får inte använda `File.WriteAllText`: filen ska öppnas (eller skapas), raderna ska skrivas ut en efter en, och filen ska slutligen stängas.

UPPGIFT 6 (6 POÄNG)

Presentföretaget Pryl och Pråal AB säljer "överraskningslådor". Varje låda kan innehålla upp till fyra mindre lådor, som i sin tur kan innehålla upp till fyra mindre lådor, och så vidare. En låda kan vara antingen svart eller vit. När man köper en låda får man inte veta hur många lådor den innehåller eller vilka färger de har. För sin egen lagerhållning behöver dock företaget hålla reda på, för varje låda, hur många "underlådor" den innehåller och vad varje underlåda har för färg.

a) Deklarera en datatyp i F# som representerar en "överraskningslåda" med alla dess underlådor! *(2p)*

b) Deklarera en funktion som tar representationen av en låda och räknar hur många vita lådor den totalt innehåller (inklusive den själv, om den är vit). *4p)*

UPPGIFT 7 (4 POÄNG)

Bevisa att följande algebraiska lag gäller: För alla listor `xs` gäller att

$$\texttt{xs} \mathbin{@} [\,] = \texttt{xs}$$

(Det vill säga: $[\,]$ är *höger enhetselement* till operatorn @. Jämför med lagen $x + 0 = x$ inom aritmetiken.)

Ledning: använd induktion över listor.

Lycka till! *Björn*