TENTAMEN I DVA 229 FUNKTIONELL PROGRAMMERING MED F#

Torsdagen den 18 augusti 2016, 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 torsdagen den 8 september 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 (6 POÄNG)

a) Deklarera en funktion `remove x l`, där `l` är en lista, som returnerar en lista där alla förekomster av `x` i `l` har tagits bort. Exempelvis ska gälla att `remove 5 [1;5;7;5]` = `[1;7]`. Använd direkt rekursion! *(4p)*

b) Samma fråga, men implementeringen ska istället göras med någon eller några av de inbyggda högre ordningens funktionerna i F#. *(2p)*

UPPGIFT 2 (2 POÄNG)

Låt funktionen `sumfirst` definieras av:

```
let rec sumfirst l =
  match l with
  | []        -> 0
  | (x,y)::xs -> x + sumfirst xs
```

Vad är den mest generella typen för `sumfirst`? Du behöver inte göra en typhärledning, men ditt svar ska innehålla en kort motivation.

UPPGIFT 3 (4 POÄNG)

Deklarera en funktion `string2list` som tar en sträng som argument och returnerar motsvarande lista av tecken (`char`)! Din lösning får inte använda någon av de inbyggda konverteringsfunktionerna i F#.

UPPGIFT 4 (3 POÄNG)

Antag att följande sekvens av deklarationer och evaluering av uttryck görs i F# interactive:

```
let y = lazy (printf "abc\n";13 + 44);;
y.Force();;
y.Force();;
```

Vad får `y` för typ? Vad händer när man evaluerar `y.Force()` första pch andra gången och vad returneras för värden?

UPPGIFT 5 (4 POÄNG)

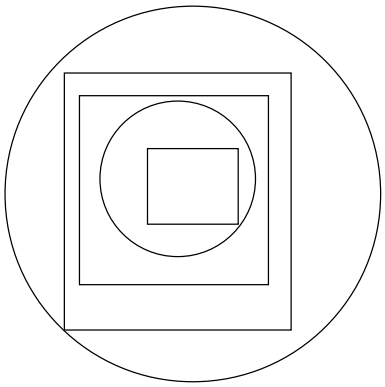
Deklarera en funktion som gör följande saker:

- tar ett filnamn `f` (sträng), en annan sträng `s` och ett heltal `n` som argument,
- öppnar filen `f` för skrivning,
- skriver strängen `s` på ny rad `n` gånger i `f`, samt
- stänger `f`.

Din lösning ska använda explicit rekursion.

UPPGIFT 6 (7 POÄNG)

En presentfirma säljer askar. En ask kan antingen vara cylindrisk eller kvadratisk. Man vill nu sälja presentkit med en mängd askar som är packade i varandra. Ett sådant kit kan innehålla både kvadratiska och cylindriska askar och de olika typerna av askar behöver inte ligga åtskilda utan en kvadratisk ask kan vara packad i en cylindrisk och vice versa. Figuren nedan visar ett exempel hur det kan se ut:



Givet en mängd askar, i en viss ordning, vill man nu kontrollera att de verkligen går att packa i varandra i den ordningen. För att en ask ska kunna packas i en annan måste två villkor vara uppfyllda:

1. den inre asken måste få plats geometriskt i den yttre, samt
2. höjden på den inre asken måste vara mindre än höjden på den yttre (för annars går det inte att sätta på locket på den yttre asken).

a) Deklarera en datatyp med vars hjälp du kan representera en mängd askar som ligger i en viss ordning! *(2p)*

b) Deklarera en funktion som tar en representation av en följd askar, i datatypen deklarerad i *a)*, och kontrollerar om askarna verkligen kan packas i den ordningen! *(5p)*

Ledning: för deluppgift *a)*, tänk igenom vilka geometriska parametrar hos askarna som behöver representeras i datatypen. För deluppgift *b)*, fundera på vilka olika fall som kan uppstå och vilka geometriska villkor som måste vara uppfyllda i de olika fallen.

UPPGIFT 7 (4 POÄNG)

Visa att

```
List.map f >> List.tail = List.tail >> List.map f
```

Du behöver inte göra ett fullt formellt bevis med induktion. Det räcker om du gör ett mer informellt bevis i den stil som vi gått igenom på föreläsning.

Lycka till! *Björn*