TENTAMEN I DVA 229 FUNKTIONELL PROGRAMMERING MED F#

Torsdagen den 13 augusti 2020, kl 14:10 – 18:30

Detta är en hemtentamen. Därför gäller lite andra regler än vanligt. Samarbete med eller hjälp från andra personer är *inte* tillåtet. I övrigt är alla hjälpmedel tillåtna (VisualStudio, MSDN, kursmaterial, etc.) Man får dock inte basera sin lösning på kod som är plankad från t.ex. nätet (OBS att vi kan kontrollera t.ex. med urkund). Man får heller inte göra sina lösningar tillgängliga för andra, t.ex. genom att lägga upp dem på github. Tentan lämnas ut via Canvas 14:30 och inlämning sker via Canvas i form av en pdf-fil innan 18:30 den 13 augusti. För godkänt krävs 15 poäng, max är 30 poäng. Resultatet offentliggörs senast torsdagen den 3 september 2020.

Vänligen observera följande:

- Om inte annat sägs ska lösningarna vara rent funktionella, dvs. sånt som muterbara variabler och sidoeffekter får inte förekomma.
- 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.
- Ge klara och tydliga förklaringar! Oklara och röriga förklaringar kan ge avdrag.
- Märk dina lösningar med namn och personnummer. (P.g.a. omständigheterna går det inte att ha en anonym tenta.)
- Endast en uppgift på ett och samma sida. Markera uppgiftsnumret tydligt. Se också till att sidorna är numrerade.
- Uppgifterna är inte nödvändigtvis sorterade i svårighetsgrad. Om du kör fast kan det löna sig att gå vidare till nästa uppgift.
- Lösningar som inkommer efter 18:30 kommer inte att bli rättade. Undantag kommer bara att göras om orsaken är något som helt klart ligger utanför er kontroll, t.ex. om nätet går ned.

Frågor på själva tentan: Björn Lisper på 0739-607199. Tekniska frågor (Canvas mm): Jean Malm på 070-9658574.

UPPGIFT 1 (6 POÄNG)

I den linjära algebran definieras skalärprodukten av två n-dimensionella vektorer $(a_1, \ldots, a_n)$, $(b_1, \ldots, b_n)$ som

$$\sum_{i=1}^{n} a_i \cdot b_i$$

a) Deklarera en funktion som tar två vektorer, representerade av listor av flyttal (`float list`) och beräknar deras skalärprodukt! Din lösning ska använda direkt rekursion.

Förklara i detalj hur din lösning fungerar, speciellt rekursionen! Ange också vad din funktion får för typ, och motivera varför funktionen får den typen. *(4p)*

b) Gör en alternativ deklaration av funktionen i *a)* som använder sig av någon eller några inbyggda högre ordningens listfunktioner i F# på ett vettigt sätt. Förklara hur din lösning fungerar och framförallt hur de högre ordningens funktionerna används! Du behöver inte ha någon speciell felhantering i fallet att listorna är olika långa. *(2p)*

UPPGIFT 2 (3 POÄNG)

Betrakta följande funktionsdeklaration:

```
let rec sum_some l = match l with
                     | [] -> 0
                     | Some x :: xs -> x + sum_some xs
                     | None :: xs -> sum_some xs
```

a) Förklara i ord vad funktionen `sum_some` gör. *(1p)*

b) Vad får `sum_some` för typ? Motivera! Du behöver inte göra en fullständig typhärledning, men en god motivering krävs för full poäng. *2p)*

UPPGIFT 3 (8 POÄNG)

Följande rekursionsekvation definierar en talföljd:

$$\begin{aligned} f_0 &= 1 \\ f_n &= 2 \cdot f_{n-1}, \quad n > 0 \end{aligned}$$

Du ska nu deklarera tre F#-funktioner som på olika sätt beräknar och returnerar f_n:

a) En funktion som använder direkt rekursion. *(2p)*

b) En funktion som använder ackumulerande argument. *(3p)*

c) En funktion som istället för ackumulerande argument använder muterbara referensceller. *(3p)*

Förklara i detalj hur dina lösningar fungerar. För deluppgift *c)* lista också de funktioner och operatorer i din lösning som opererar på muterbara referensceller och förklara för var och en av dem vad den operatorn/funktionen gör. Du behöver inte ha någon speciell felhantering av fallet $n < 0$.

UPPGIFT 4 (3 POÄNG)

I kursen har vi gått igenom vad lat och ivrig evaluering är (call by need, call by value). F# har call by value, men skulle lika gärna kunna ha haft call by need. Deklarera en funktion som beter sig annorlunda beroende på om anrop till den räknas ut med call by need eller call by value! Förklara också varför din funktion har den egenskapen! Ge ett exempel på där din funktion används på ett sätt som gör att den uppför sig annorlunda beroende på vilken sorts evaluering som används, och förklara varför det blir så!

UPPGIFT 5 (6 POÄNG)

Tågoperatörer kan behöva representera lok, vagnar och tåg i sina system. Det är då naturligt att skapa datatyper som stöder detta. I denna uppgift ska vi representera en förenklad modell där ett lok har attributen vikt (kg, flyttal), motoreffekt (kW, flyttal), samt typbeteckning (sträng), en vagn har attributen vikt och antal passagerarplatser och ett tåg består av ett lok + ett antal vagnar. OBS att ett ensamt lok också utgör ett tåg.

a) Deklarera en F#-datatyp för att representera lok, vagnar och tåg enligt ovan! Förklara hur din representation fungerar.

b) Man kan vilja kontrollera att ett tåg inte riskerar att ha för klen motoreffekt i förhållande till dess totala vikt. För att göra detta kan man räkna ut kvoten mellan lokets motoreffekt och tågets maximala vikt (kW/kg). Deklarera en F#-funktion (eller flera funktioner) som gör detta för ett tåg representerat i din datatyp! Du får anta att en passagerare i genomsnitt väger 80 kg. Förklara klart och tydligt hur din funktion fungerar.

UPPGIFT 6 (4 POÄNG)

Betrakta följande funktionsdeklaration:

```
let rec f x y = if x then y else f x y
```

Härled en mest generell typ för `f`! För full poäng krävs en detaljerad redovisning för typhärledningen.

Lycka till! *Björn*