TENTAMEN I DVA 229 FUNKTIONELL PROGRAMMERING MED F#

Tisdagen den 9 juni 2020, kl 14:10 – 19:30 (OBS en extra timme!)

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). Tentan lämnas ut via Canvas 14:30 och inlämning sker via Canvas i form av en pdf-fil innan 19:30 den nionde juni. För godkänt krävs 15 poäng, max är 30 poäng. Resultatet offentliggörs senast tisdagen den 30 juni 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.
- P.g.a. situationen med en provisorisk tentamensform, som kanske kan göra det jobbigare att skriva tentan, får ni en timme extra på er. Men inte mer! Lösningar som inkommer efter 19:30 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.
- Lösningsförslag kommer att finnas på kursens hemsida efter att tentan är slut.

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

UPPGIFT 1 (6 POÄNG)

a) Deklarera en funktion som tar en lista av heltal (`int`) och returnerar summan av alla tal i listan som är strikt större än noll! Använd 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! *(2p)*

UPPGIFT 2 (4 POÄNG)

Följande deklaration är ett möjligt sätt att summera elementen i en array av flyttal:

```
let rec asum a = match a with
                 | [||] -> 0.0
                 | _    -> a.[0] + asum (a.[1..])
```

a) Förklara vad det är för mönster i match-uttrycket! Vad matchar de för något? *(1p)*

b) Denna deklaration har ett problem. Vilket? Förklara! En god motivering krävs för full poäng. *3p)*

UPPGIFT 3 (3 POÄNG)

Följande är en utskrift från en session med F#-tolken (F# Interactive 4.0), där användaren matar in tre kommandon (de rader som avslutas med "`;;`") och får tre svar:

```
> let x = lazy (printf "Hej!\n";[3;2;1]);;

val x : Lazy<int list> = Value is not created.

> x.Force();;
Hej!
val it : int list = [3; 2; 1]
> x.Force();;
val it : int list = [3; 2; 1]
```

Förklara i detalj vad som händer och varför svaren från F#-tolken ser ut som de gör!

UPPGIFT 4 (4 POÄNG)

Deklarera en funktion som tar en array av flyttal (`float`) och returnerar medelvärdet av elementen i arrayen! Summeringen av arrayens element ska ske med direkt rekursion, med hjälp av en muterbar referenscell som successivt ackumulerar summan. (Dvs. du får i denna uppgift *inte* använda inbyggda standardfunktioner som `Array.sum`.) Förklara i detalj hur din lösning fungerar. 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.

UPPGIFT 5 (3 POÄNG)

Här är två snarlika fall där två funktioner med samma namn `f` deklareras:

```
// Fall 1:
let f x = x;;
let rec f x = if x = 0 then 1 else x + f (x-1)

// Fall 2:
let f x = x;;
let f x = if x = 0 then 1 else x + f (x-1)
```

a) Räknar man ut `f 5` får man i det första fallet svaret `16` och i det andra fallet `9`. Förklara varför det blir olika svar i de två fallen! *(2p)*

b) Säg nu att vi ändrar ordningen på deklarationerna i det andra fallet:

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

Försöker vi nu kompilera denna kod får vi ett felmeddelande att `f` inte är definierad. Förklara varför! *(1p)*

UPPGIFT 6 (6 POÄNG)

Inom datorgrafiken används ofta olika sorts träd för att representera volymer. En variant är ett träd där varje intern nod har precis 8 söner och varje löv representerar någon enkel volym, t.ex. en sfär. Ett sådant träd kan då approximera volymen hos någon geometrisk form. Sådana träd kan användas t.ex. för kollisionsdetektering.

a) Deklarera en F#-datatyp för träd enligt ovan, där varje löv representerar en sfär vars radie ges av ett flyttal (`float`) och där koordinaterna för dess mittpunkt ges av tre flyttal! Förklara hur din datatyp representerar träden ovan. Är din datatyp polymorf eller ej? *(2p)*

b) Deklarera en funktion som räknar ut volymen för ett träd enligt ovan representerat av din datatyp! Du får anta att inga sfärer överlappar. Förklara klart och tydligt hur din funktion fungerar och framförallt hur trädet traverseras. *(4p)*

Ledning: volymen för en sfär med radien r är $4\pi r^3/3$.

UPPGIFT 7 (4 POÄNG)

Betrakta lambda-uttrycket

$$(\lambda z.(\lambda y.z))\ ((\lambda x.x)\ x)$$

β-reducera uttrycket i alla möjliga ordningar! Visa reduktionerna och slutresultatet. *(4p)*

Lycka till! *Björn*