Försättsblad till skriftlig tentamen vid Linköpings Universitet

Duggan skall EJ ske anaonymt

Datum för tentamen/Dugga	2010-03-04
Sal (2) Om tentan går i flera salar ska du bifoga ett försättsblad till varje sal och ringa in vilken sal som avses	TER1 TER2
Tid	8-10
Kurskod	TDDC74
Provkod	TEN1 (dugga 3)
Kursnamn/benämning **Provnamn/benämning**	Programmering - abstraktion och modellering Skriftlig tentamen/duggor
Institution	IDA
Antal uppgifter som ingår i tentamen	4
Jour/Kursansvarig Ange vem som besöker salen	Anders Haraldsson
Telefon under skrivtiden	281403 (arb), 0705 147709 (mobil)
Besöker salen ca kl.	kl 9
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ank 2362, annek@ida.liu.se
Tillåtna hjälpmedel	Inga

Tekniska högskolan vid Linköpings universitet
Institutionen för datavetenskap
Anders Haraldsson

TDDC74 Programmering, abstraktion och modellering

DUGGA 3

Torsdag 4 mars 2010 kl 8-10

Namn: _____

Personnummer: _____

Skriv även ditt namn på varje uppgiftssida.

Uppgifterna löses direkt på denna uppgiftslapp, som lämnas in i sedvanligt tentamensomslag.

Skriv tydligt så att inte dina lösningar missförstås. Använd väl valda namn på parametrar etc.

Skriv lagom stort, en del skriver så smått så att vi har svårt att tolka texten och en del skriver så stort så at det inte ryms på raden. Börja raden så långt till vänster som möjligt.

Även om det i uppgiften står att du skall skriva *en* funktion, så får du gärna skriva ytterliggare hjälpfunktioner, som kan vara nödvändiga.

På uppgifterna kan halva poäng utdelas.

Betygsgradering: Det är tre duggor. Varje dugga ger 12p, dvs totalt 36p. För att passera en dugga krävs minst 3p på duggan. Totalt skall du på de tre duggorna för betyget 3 ha minst 20p. För betyget 4 minst 25p och för betyget 5 minst 30p.

Om du har tid: Denna dugga kan ge extrapoäng, som räknas in i totalpoängen.
Se uppgift 3.

Lycka till

Uppgift 2. Omgivningsdiagram (4 poäng)

2a. (2p) Förklara följande begrepp i samband med omgivningsdiagram:

- bindning (*binding*)

- ram (*frame*)

- omgivning (*environment*)

Förklara sedan hur omgivningsdiagrammet används då man i Scheme ger följande uttryck:

- Definition: `(define x ...)` i en omgivning *En*

- Tilldelning: `(set! x ...)` i en omgivning *En*

- Lambda-uttryck: `(lambda` *`formella-parametrar kropp`*`)` i en omgivning *En*

- Proceduranrop/procedurapplikation: `(procedur` *`aktuella-parametrar`*`)` i en omgivning *En*

Uppgift 3. Procedurobjekt med tillstånd (2p)

Vi önskar en funktion `(fib n)`, som beräknar fibonacci-talen enligt följande

$fib_0 = 0$
$fib_1 = 1$
$fib_n = fib_{n-1} + fib_{n-2}$

Funktionen skall minnas tidigare beräknade fibonaccital för att inte behöva räkna om samma fibonaccital mer än en gång. Detta kan göras så att man har en associationslista, där man lagrar n, som nyckel och fib_n, som värde.

Komplettera nedanstående definition av `fib`.

```
(define fib
  (let ((tidigare-värden '()))
    (lambda (n)
      (let ((n-värde-par (assq[1] n tidigare-värden)))
        (if (pair? n-värde-par)
            (cdr n-värde-par)        ; tidigare beräknat
            ; komplettera med kod här
```

```
)))))
```

Extrauppgift (1p): Gör ett omgivningsdiagram, som visar hur `fib` är definierad och vad som händer då du beräknar `(fib 3)`. Använd sid 8 om inte platsen här räcker till.

[1] Proceduren `assq` returnerar det par i en associationslista, som har en given nyckel. Annars returneras #f.

Uppgift 4. Objektorienterad modellering, forts

4a. (1p) Skriv Scheme-uttryck, som ger de tre sakerna en färg.

4b. (1p) Skriv Scheme-uttryck, som lägger in de tre sakerna i "Lisa".

4c. (1p) Komplettera koden för `leksaker-färg`, som ger en lista med namnen på de leksaker som har en given färg. Vi kan då t. ex. få reda på att "dockan-lisa" och "tennisboll" är gula.