Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2013-02-07
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	KTR1
Kursnamn/benämning **Provnamn/benämning**	Programmering - abstraktion och modellering Frivillig dugga
Institution	IDA
Antal uppgifter som ingår i tentamen	5
Jour/Kursansvarig Ange vem som besöker salen	Jalal Maleki
Telefon under skrivtiden	ankn. 19 63 eller 070-607 19 63
Besöker salen ca kl.	kl. 08:45
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Vilken typ av papper ska användas, rutigt eller linjerat	Valfritt
Antal exemplar i påsen	

TDDC74 Programmering: Abstraktion och modellering
Dugga 1 , kl 8-10, 7 feb 2013

Läs alla frågorna först och bestäm dig för den ordning som passar dig bäst. Skriv tydligt för att öka läsbarheten. Använd **väl valda namn** på parametrar och **indentera** din kod.

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

Betygsgradering: Det finns tre duggor i kursen. Varje dugga ger 12p, dvs totalt 36p för alla tre. För att passera en dugga krävs minst 3p på duggan. Totalt skall du på de tre duggorna för betyget 3 ha minst 18p, för betyget 4 minst 23p och för betyget 5 minst 27p.

Lycka till!

Uppgift 1

(3 poäng) Antag att vi har evaluerat följande uttryck I Scheme.

```
(define x (+ 2 1)
(define y (* 2 (+ x 1))
(define square (lambda (x) (* x x)))
(define double (lambda (x) (+ x x)))
(define f (lambda (g)
            (lambda (x)
               (g (square (double x))))))
```

Vad blir värdet av följande uttryck?

```
1. y
2. (+ x y)
3. (square (double x))
4. (> y x)
5. (lambda (x y) (- x y))
6. ((f double) 1)
```

OBS! I fall det värde som returneras som resultat är en procedur, kan du skriva t ex att "Värdet är en procedur ..." och sedan ange vilka parametrar och vilken kropp proceduren har.

Uppgift 2

(3 poäng) Vad blir värdet av följande uttryck om det evalueras i Scheme?

```
(let ((a 17))
  (let ((a 3)
        (b a))
    (+ a (- b 7))))
```

Skriv sedan om uttrycket så att **let**-uttrycken ersätts med motsvarande **lambda**-uttryck.

Uppgift 3

(3 poäng) Skriv Schemeproceduren **binary->decimal** som tar ett binärt tal och konverterar det till dess motsvarande decimalt tal. Följande interaktion visar exempel på hur detta skall fungera.

```
> (binary->decimal 110)
6
> (binary->decimal 0)
0
> (binary->decimal 1111)
15
> (binary->decimal 10000)
16
```

Uppgift 4

(3 poäng) Följande function (**perfekt**?) tar ett positivt heltal som argument och returnerar **#t** om heltalet är ett perfekt tal, annars **#f**. Perfekta tal är sådana som är hälften så stora som summan av sina delare, t ex, 6 är hälften av summan 1+2+3+6 där 1, 2, 3 och 6 räknas som dess delar. Nästa perfekta tal är 28 som är hälften av 1+2+4+7+14+28.

```
(define perfekt?
  (lambda (n)
    (define (delare? i n)
        (= (remainder n i) 0)))
    (define (summera-delare i)
      (cond ((> i n) 0)
            ((delare? i n)
             (+ i (summera-delare (+ i 1))))
            (else (summera-delare (+ i 1)))))
    (= (summera-delare 1) (* 2 n))))
```

Skriv en procedur med namnet **perfekt-efter** som tar ett heltal **n** och returnerar det minsta perfekta talet som är större än **n**. Följande interaktion visar exempel på hur detta skall fungera. (De första perfekta talen är 6, 28 och 496.)

```
> (perfekt-efter 5)
6
> (perfekt-efter 6)
28
> (perfekt-efter 12)
28
> (perfekt-efter 100)
496
```