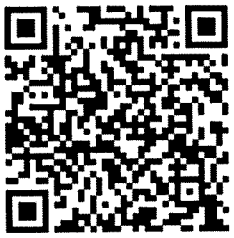
Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-04-07
Sal (3)	TER3 TER4 TERE
Tid	8-10
Kurskod	TDDC74
Provkod	TEN1
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 Märak-Leffler (kl. 8-10) Jalal Maleki (kl. 10-11)
Telefon under skrivtiden	Anders (0731-011291), Jalal (0706-071963)
Besöker salen ca klockan	ca. 08:45
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-04-07
Sal (3)	TER3 TER4 TERE
Tid	8-10
Kurskod	TDDC74
Provkod	TEN1
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 Märak-Leffler (kl. 8-10) Jalal Maleki (kl. 10-11)
Telefon under skrivtiden	Anders (0731-011291), Jalal (0706-071963)
Besöker salen ca klockan	ca. 08:45
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet



Datum för tentamen	2016-04-07
Sal (3)	TER3 TER4 TERE
Tid	8-10
Kurskod	TDDC74
Provkod	TEN1
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 Märak-Leffler (kl. 8-10) Jalal Maleki (kl. 10-11)
Telefon under skrivtiden	Anders (0731-011291), Jalal (0706-071963)
Besöker salen ca klockan	ca. 08:45
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

TDDC74 Programmering: Abstraktion och modellering
Dugga 3, kl 8—10, 7 april 2016

Läs alla frågorna först och bestäm dig för i vilken ordning du vill lösa uppgifterna. Uppgifterna är inte ordnade i någon särskild ordning!

Skriv tydligt och läsligt. Använd **väl valda namn** på parametrar och **indentera** din kod. Väl valda namn inkluderar konsekvent språk med mera.
Stora parenteser får användas för att avsluta alla öppna parenteser.

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

OBS! I denna dugga får du använda alla primitiver, inklusive kommandon som `set!`, `set-mcar!`, `set-mcdr!` för att ändra bindningar och ändra i strukturer.

Betyg: Det finns tre duggor i kursen. På varje dugga kan man få som mest 12p (så totalt 36p för alla tre). För att kunna få betyget 3 på duggorna måste du sammanlagt (på alla tre duggor) ha fått minst 18p, för betyget 4 gäller minst 23p och för betyget 5 minst 27p.

Lycka till!

Uppgift 1, Omgivningsdiagram (3 poäng)

Rita det omgivningsdiagram som uppstår då de följande uttrycken evalueras. Alla evalueringar skall visas i ett och samma omgivningsdiagram. När värdet på en variabel ändras, kryssa över det gamla värdet och skriv det nya bredvid.

```
(define x 42)

(set! x (+ x 6))

(define f
  (lambda (n)
    (* n 2)))

((lambda (n) (* n 2))
 17)

(f 100)

(let ((state 0))
  (lambda ()
    (set! state (- 1 state))))
```

Uppgift 2, 'box-pointer'-diagram (2 poäng)

Rita box-pointer-diagram över det som händer när följande kod evalueras:

```
(define a (mcons 1 (mcons 2 3)))
(define b (mcons 4 5))
(set-mcdr! (mcdr a) b)
(set! b (mcons 6 7))
```

Om något förändras, ska det framgå hur det förändras. Stryk över gamla pilar och rita nya (snarare än att rita helt nya kopior av hur det ser ut efter förändringen).

Uppgift 3, Tillståndsmodellering (3 poäng)

Vi vill representera ett minne `mem` för siffror, där varje minne ska ha koll på just sitt värde. Det ska hantera kommandona `add` och `sub`, vilket ökar respektive minskar minnets värde med indata och returnerar minnets värde efter ändringen. Så här ska det fungera:

```
> (define m1 (make-mem))
> (define m2 (make-mem))

> (m1 'add 10)
10

> (m2 'add 0)
10

> (m1 'sub 5)
5

> (m1 'add 2)
7
```

Uppgifter

a) Skriv `make-mem`. (2p)

b) Skriv en procedur `reset-value` som tar ett minne och nollställer dess värde. (1p)

```
> (m1 'add 0)
7
> (reset-value m1)
> (m1 'add 0)
0
```

Uppgift 4, Dataabstraktion och programmering (4 poäng)

I denna uppgift väljer vi att beskriva en digitalklocka. En klocka kommer fysiskt att sparas i muterbara cons-celler. Vår digitalklocka visar tiden som två tal: timmar och minuter. Minuter varierar i intervallet 0-59 och timmar i 0-23 (kl 00:08, t ex, beskrivs som {0 8}).
OBS! Läs igenom hela uppgiften. Deluppgifterna kan lösas oberoende av varandra, dvs, i del b) kan du anta att procedurerna i del a) finns och fungerar som de skall.

Konstrukorproceduren för en klocka definierar vi så här:

```
(define (make-clock hours minutes)
  (mcons hours (mcons minutes '())))
```

Uppgifter

a) Komplettera dataabstraktionen genom att implementera följande selektorer och mutatorer. (1 poäng)

```
(define (clock-hours clock)
  ...)

(define (clock-minutes clock)
  ...)

(define (clock-set-hours! clock hours)
  ...)

(define (clock-set-minutes! clock minutes)
  ...)
```

Allt mutatorerna behöver göra är att sätta rätt del av strukturen till det angivna värdet. De behöver inte göra någon kontroll av indata. T ex behöver `clock-set-hours!` inte testa att $0 \leq hours < 23$.

b) Skriv proceduren `tick` som tickar fram en klocka med en minut. Nedan skapar vi två klockor och får dem att ticka ett par gånger. (3 poäng)

```
> (define clock-1 (make-clock 12 58))
{12 58}

> (define clock-2 (make-clock 23 59))
{23 59}
```

```
> (tick clock-1)

> clock-1
{12 59}

> (tick clock-1)

> clock-1
{13 0}

> clock-2
{23 59}

> (tick clock-2)

> clock-2
{0 0}
```