Försättsblad till skriftlig tentamen vid Linköpings Universitet

(fylls i av ansvarig)

Datum för tentamen	*2010-01-11*
Sal	*TER1*
Tid	*14-18*
Kurskod	*TDDC74*
Provkod	*TEN1*
Kursnamn/benämning	*Programmering - abstraktion och modellering*
Institution	*IDA*
Antal uppgifter som ingår i tentamen	*7*
Antal sidor på tentamen (inkl. försättsbladet)	*6*
Jour/Kursansvarig	*Anders Haraldsson*
Telefon under skrivtid	*281403 /0705-147709*
Besöker salen ca kl.	
Kursadministratör (namn + tfnnr + mailadress)	*Anna Grabska Eklund Ankn. 23 62, annek@ida.liu.se*
Tillåtna hjälpmedel	*inga*
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	
Vilken typ av papper ska användas, rutigt eller linjerat	
Antal exemplar i påsen	

Tekniska högskolan vid Linköpings universitet
Institutionen för datavetenskap
Anders Haraldsson

TDDC74 Programmering, abstraktion och modellering

Tentamen

Måndag 11 januari 2010 kl 14-18

Uppgifterna löses direkt på denna uppgiftslapp, som lämnas in i sedvanligt tentamensomslag. Räcker inte platsen kan du använda sista sidan eller komplettera med löst blad.

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

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. Totalt kan 24p erhållas.

Betygsgradering:

12,0-16,0	betyg 3
16,5-20,5	betyg 4
21,0-24,0	betyg 5

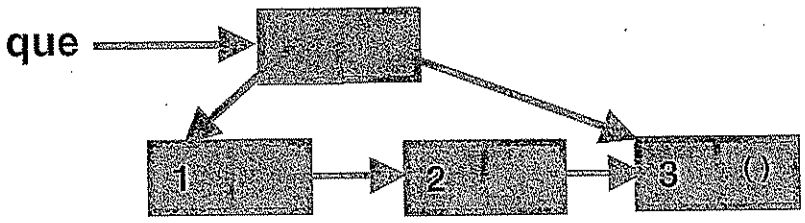
Lycka till

Uppgift 2. Listor, cons-celler och pekare (3 poäng)

2a Vad blir värdet i parentesformat av följande uttryck. Rita även upp värdet med den grafiska representationen med cons-celler och pekare (box and pointer notation).

```
(define test (cons (cons 'x '()) (list 'a 'b)))
test =
```

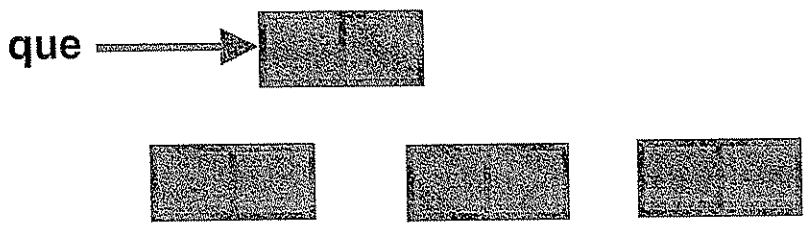
2b. En kö representeras med ett köhuvud och en lista med köelement. Köhuvudet anger första resp. sista köelementet. Skriv ett Scheme-uttryck som skapar följande köstruktur:



2c. Vad händer efter det att följande två Scheme-uttryck har beräknats.

```
(set-car! (cdr  que) 4)
(set-cdr! (car que) (cdr (cdr (car que))))
```

Rita in pekare och värden hur strukturen nu ser ut:



Uppgift 4. Rekursiva procedurer över listor/sekvenser (4 poäng)

4a. Skriv en rekursiv funktion (ta-bort-första n lista), som i en lista tar bort de *n* första elementen. Finns inte antalet element att ta bort, returneras tomma listan. Funktionen skall göra en *rekursiv processlösning*.

(ta-bort-första 3 '(a b c d e f)) = (d e f)
(ta-bort-första 10 '(a b c d e f)) = ()

(define (ta-bort-första n lista)

4b. Skriv en rekursiv funktion (spara-första n lista), som i en lista skapar en ny lista med de n första elementen. Om n är större än antalet element i listan sparas hela listan. Funktionen skall göra en *iterativ processlösning*.

(spara-första 3 '(a b c d e f)) = (a b c)
(spara-första 10 '(a b c d e f)) = (a b c d e f)

(define (spara-första n lista)

Uppgift 6. Procedurobjekt och omgivningsdiagram (4 poäng)

6a. Följande Scheme-uttryck beräknas. Vad blir värdet av sista uttrycket. Rita hur omgivningsdiagrammet ser ut efter det att alla uttrycken beräknats.

```
(define x  5)
(define y 10)
(define f (lambda (x) (+ x y)))
(define g (lambda (y) (f (+ y 1))))
(g 3) =
```

6b. Samma i denna uppgift. Vilka värden skrivs ut och rita omgivningsdiagrammet.

```
(define a '())

(define memory
        (let ((mem 0))
          (lambda (x) (set! mem (- mem x)) (set! a (cons x a)))))

(memory 20) =
(memory 10) =
a =
```

Vi definierar följande konstruktorer:

```
(define (skapa-student namn pnr) (märk 'student (list namn pnr))
(define skapa-kurs (kod kurs-namn) (märk 'kurs (list kod kurs-namn)))
(define skapa-resultat (pnr kod) (märk 'resultat (list pnr kod)))
```

Vi önskar följande 4 selektorer:

```
(define (pnr obj)  ; för både student och resultat
```

Komplettera här med kod

```
(define (namn obj)  ; för student
```

Komplettera här med kod

```
(define (kod obj)  ; för kurs och resultat
```

Komplettera här med kod

```
(define (kurs-namn obj)  ; för kurs
```

Komplettera här med kod

Vi skall kunna t ex kunna göra följande:

```
(pnr (skapa-student '(Anders Andersson) '820202-0202)) = 820202-0202
(pnr (skapa-resultat '800202-0202 'TDDX00)) = 820202-0202
(kurs-namn (skapa-resultat '800202-0202 'TDDX00)) = TDDX00
```

Till sist vill vi ha en procedur (plocka-ut index db fn), som givet ett index och en databas finner den lagrade datalistan och ur den plockar ut alla poster som uppfyller funktionen *fn*. I nedanstående exempel vill vi plocka ut alla poster i student-listan för studenter som heter Anders i förnamn.

```
(plocka-ut 'studenter Y-programmet (lambda (e) (eq (car (namn e)) 'anders))) =
= ((student (anders andersson) 800202-0202) (student (anders eriksson) 800404-0404))

(define (plocka-ut index db fn)
```

Definiera proceduren här