Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2012-03-28
Sal (2) Om tentan går i flera salar ska du bifoga ett försättsblad till varje sal och ringa in vilken sal som avses	TER3 TER4
Tid	14-16
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	Jalal Maleki
Telefon under skrivtiden	070-607 19 63
Besöker salen ca kl.	ca. 14:30
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	
Antal exemplar i påsen	

AID-nummer:	Datum: 2012-03-28
Kurskod: TDDC74	Provkod: KTR2

TDDC74 Programmering: Abstraktion och modellering
Dugga 3

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.

Det finns tre uppgifter i denna dugga och varje uppgift har 4 poäng.

Betygsgradering: Det finns tre duggor i kursen. 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 19p, för betyget 4 minst 24p och för betyget 5 minst 29p.

Lycka till!

AID-nummer:	Datum: 2012-03-28
Kurskod: TDDC74	Provkod: KTR2

Uppgift 1

Skriv en Schemeprocedur `(remember-all n)` som tar ett argument *n* och som returnerar en lista som innehåller alla argument som proceduren har anropats med hittills. Se följande exempel:

```
> (remember-all 1)
(1)
> (remember-all 2)
(2 1)
> (remember-all 9)
(9 2 1)
```

AID-nummer:	Datum: 2012-03-28
Kurskod: TDDC74	Provkod: KTR2

Uppgift 2

Rita omgivningsdiagram som fås efter de följande åtta uttrycken är evaluerade. När värdet på en variabel ändras kryssa över det gamla värdet och skriv det nya bredvid.

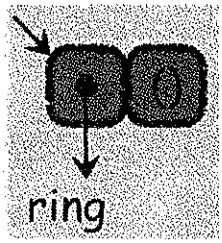
```
(define x 22)

(set! x (+ x 6))

(define f
  (lambda (n)
    (if (= n 0)
        1
        (+ 2 (f (- n 1))))))

(* (f 0) (f 2))

(define g
  (lambda (a)
    (let ((b a))
      (set! x (+ x b)))))

(g 4)
```

AID-nummer:	Datum: 2012-03-28
Kurskod: TDDC74	Provkod: KTR2

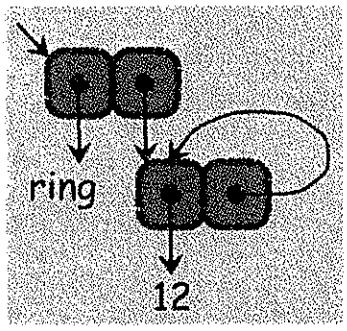
Uppgift 3

I den här uppgiften skall du jobba med en ring som definieras enligt figurerna nedan. Läs genom introduktionen nedan och gör uppgifterna (a) och (b). Observera att strukturen skapas med hjälp av muterbara CONS-celler (`mcons`).

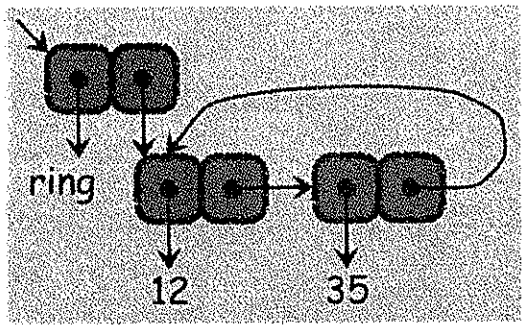
En ring med inga element.

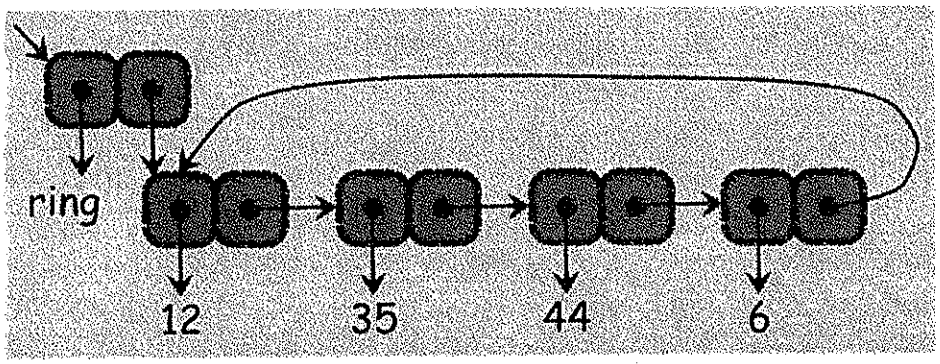


En ring med 1 element:



Två ringar med 2 resp 4 element:





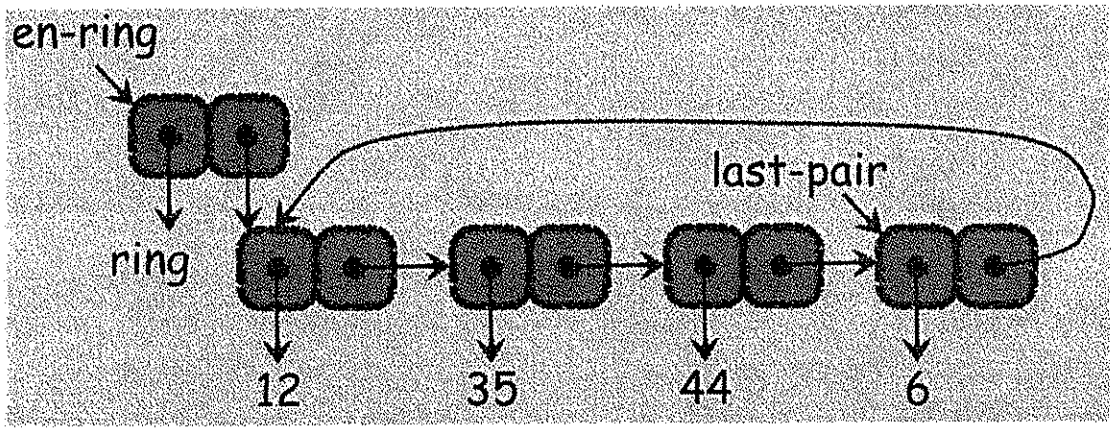
Som Ni ser från exemplen ovan, har en ring ett antal element där sista elementet pekar tillbaka till det första och på så sättet bildas en cirkulär struktur som vi kallar en ring. Vi definierar ring som en dataabstraktion. Här är några av de procedurer som ingår i implementationen av denna dataabstraktion:

```
(define (make-ring) (mcons 'ring ()))

(define (ring-label ring)
  (mcar ring))

(define (ring-first ring)
  (mcdr ring))

(define (empty-ring? ring)
  (null? (mcdr ring)))

(define (ring-set-first! ring new-element)
  (set-mcdr! ring new-element))
```

Använd dessa vid behov i samband med din lösning till uppgifterna nedan.

a) Skriv proceduren (`find-last` *`ring`*) som tar en ring som argument och returnerar dess sista element – dvs sista paret. Följande uttryck anropar denna procedur och binder `last-pair` till resultatet av anropet:

```
(define last-pair (find-last en-ring))
```



b) Skriv proceduren (`ring-insert-last!` *`ring value`*) som lägger ett element som innehåller *`value`* sist i ringen. Exempel på hur den skall kunna användas finns på nästa sida.

```
(define en-ring (make-ring))
(ring-insert-last! en-ring 12)
(ring-insert-last! en-ring 35)
(ring-insert-last! en-ring 44)
(ring-insert-last! en-ring 6)
```

Observera att dessa fyra anrop kommer att skapa ringen med fyra element som finns i den tidigare figuren.