Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2012-05-22
Sal (1) Om tentan går i flera salar ska du bifoga ett försättsblad till varje sal och ringa in vilken sal som avses	TER3
Tid	14-18
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 kl. 15
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	

AID-nummer:	Datum: 2012-05-22
Kurskod: TDDC74	Provkod: TEN1

TDDC74 Programmering: Abstraktion och modellering
Tentamen

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.

Tentan bestå av fyra uppgifter där den fjärde i sin tur har fyra deluppgifter.

Betygsgradering: För betyget 3 krävs minst 12p, för betyget 4 minst 16p och för betyget 5 minst 20p. Högst möjliga antal poäng är 24.

Lycka till!

AID-nummer:	Datum: 2012-05-22
Kurskod: TDDC74	Provkod: TEN1

Uppgift 1

(4 poäng) Schemeproceduren `(remember-all n)` tar ett argument *n* och returnerar en lista som innehåller alla argument som proceduren har anropats med hittills. Se definition och exempel på anrop nedan.

```
(define remember-all
  (let ((all-args ()))
    (lambda (arg)
      (set! all-args (cons arg all-args))
      all-args)))
```

Här är några exempelanrop till proceduren:

```
> (remember-all 5)
(5)
> (remember-all 2)
(2 5)
> (remember-all 23)
(23 2 5)
```

Rita de omgivningsdiagram som fås efter de fyra uttrycken ovan har evaluerats (dvs definitionen på `remember-all` och de tre anropen).

Uppgift 2

(4 poäng) Medianen är det tal i en mängd som storleksmässigt ligger så att det finns lika många tal som är större än och mindre än medianen. Av talen 1, 7, 9, 10 och 17 är 9 medianen (medan medelvärdet är 8,8). För mängder med ett jämnt antal tal definieras medianen som medelvärdet av de två tal som ligger i mitten.

Skriv proceduren `(median lista)` som returnerar medianen av de tal som ingår i `lista`. Du kan utgå ifrån att `lista` innehåller minst ett tal och är sorterad i ökande ordning.

```
> (median '(1 7 9 10 34))
9
> (median '(3 5 9 10 20 49))
9.5
> (median '(1 6 9))
6
> (median '(2 4))
3
> (median '(2))
2
```

AID-nummer:	Datum: 2012-05-22
Kurskod: TDDC74	Provkod: TEN1

Uppgift 3

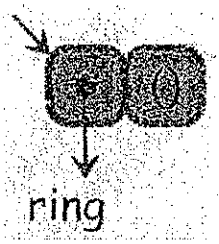
(4 poäng) Skriv om följande `let`-uttryck som ett `lambda`-uttryck:

```
(let ((a 4)
      (b 9)
      (c 2))
  (let ((disc (- (* b b) (* 4 (* a c)))))
    (/ (+ (- b) (sqrt disc))
       (* 2 a))))
```

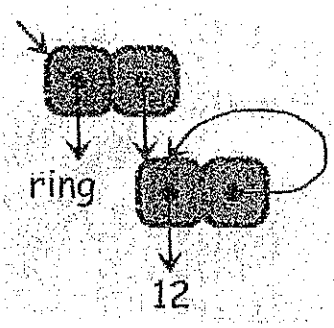
Uppgift 4

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

En ring med inga element.

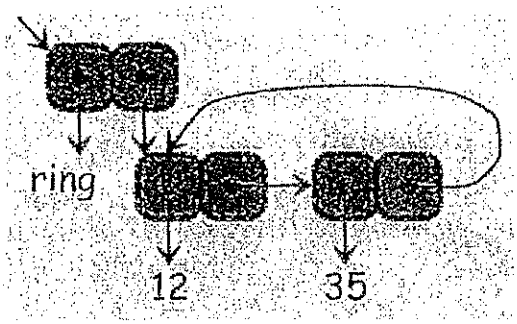


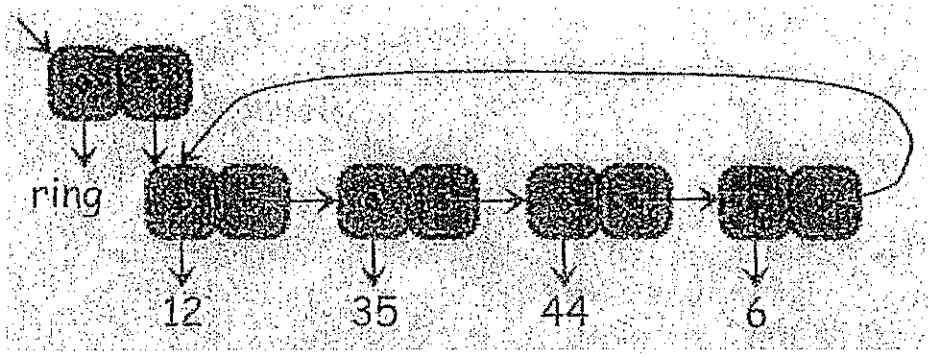
En ring med 1 element, där elementets värdeinnehåll är 12:



Nedan finns två ringar med 2 resp 4 element. Observera att det kan finnas flera lika stora värden i ringens element.

AID-nummer:	Datum: 2012-05-22
Kurskod: TDDC74	Provkod: TEN1





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

```
(define (make-ring)
   ;Skapar en ny ring
   (mcons 'ring ()))

(define (ring-label ring)
   ;Returnerar symbolen ring
   (mcar ring))

(define (ring-first ring)
   ;Returnerar
   (mcdr ring))

(define (empty-ring? ring)
   ;Har ringen inga element?
   (null? (mcdr ring)))

(define (ring-set-initial-element! ring new-element)
   ;Skapar den allra första elementet i ringen
   ;Vi skapar denna procedur för att inläggningen av
   ;det första elementet inte följer inläggning av
   ;de följande elementen.

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

AID-nummer:	Datum: 2012-05-22
Kurskod: TDDC74	Provkod: TEN1

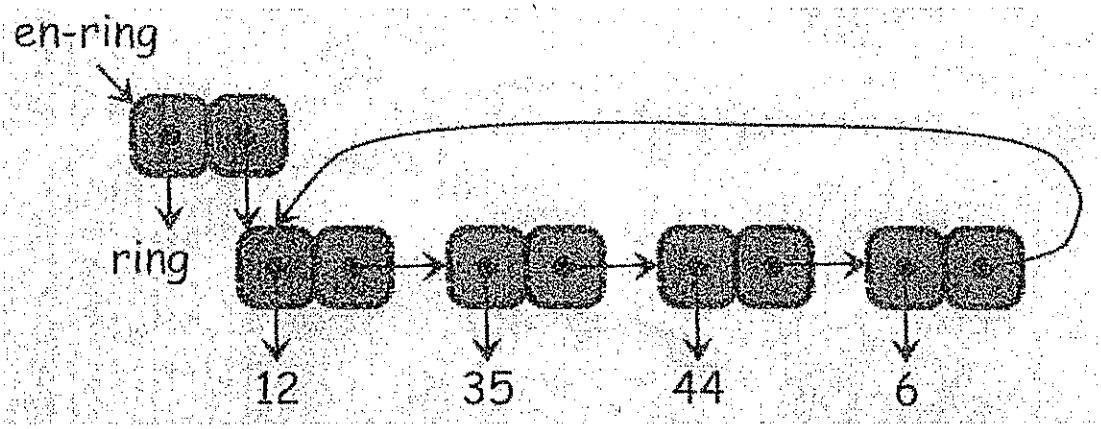
```
(define (make-new-element val)
   ;Skapar ett nytt element med värdet val
   (mcons val ()))

(define (element-value element)
   (mcar element))

(define (element-next element)
   (mcdr element))
```

Använd dessa vid behov i samband med dina lösningar till uppgifterna nedan. Komplettera gärna dessa abstrakta datadefinitionerna om du ser att flera funktioner behövs.

Här är senaste bilden igen, men vi har kallat ringen för `en-ring` och den används i samband med tentauppgifterna nedan:



1) (3 poäng) Denna första deluppgift är igentligen ingen oberoende deluppgift utan kräver att du använder dataabstraktionerna ovan på rätt sätt I de andra tre uppgifterna (3 poäng).

2) (3 poäng) Skriv proceduren (`ring-insert-first!` *`ring value`*) som lägger ett element med värdet *`value`* först i ringen. Exempel på hur den skall kunna användas följer.

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

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

AID-nummer:	Datum: 2012-05-22
Kurskod: TDDC74	Provkod: TEN1

3) (3 poäng) Skriv proceduren **(ring-count-elements** *ring*) som räknar antalet element i ringen. För ringen i den senaste figuren (kallat **en-ring**) returneras värdet 4.

```
> (ring-count-elements en-ring)
4
```

4) (3 poäng) Skriv en procedur **(ring-sum-values** *ring*) som summerar värdena som finns i ringens element. För ringen i den senaste figuren (kallat **en-ring**) returneras värdet 97 (12+35+44+6).

```
> (ring-sum-values en-ring)
97
```