Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-10-19
Sal (2)	TER1 TERD
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	6
Jour/Kursansvarig Ange vem som besöker salen	Jalal Maleki
Telefon under skrivtiden	Jalal 0706-071963
Besöker salen ca klockan	ca. 15:00
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	2016-10-19
Sal (2)	TER1 TERD
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	6
Jour/Kursansvarig Ange vem som besöker salen	Jalal Maleki
Telefon under skrivtiden	Jalal 0706-071963
Besöker salen ca klockan	ca. 15:00
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
Tentamen, onsdag 19 oktober 2016, kl 14–18

Läs alla frågorna först, och bestäm dig för i vilken ordning du vill lösa uppgifterna.

Skriv tydligt och läsligt. Använd **väl valda namn** på parametrar och **indentera** din kod.Väl valda namn omfattar exempelvis att inte blanda språk.

Om du vill avsluta *samtliga* parenteser i en funktionsdefinition, kan du om du vill skriva en stor parentes. Observera att detta innebär att du sluter *samtliga* öppna parenteser från och med sista `define` du öppnade.

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

Observera att poängavdrag kan ges för onödigt komplicerade eller ineffektiva lösningar.

Preliminära betygsgränser:

12-15	betyg 3
15.5-18.5	betyg 4
19-24	betyg 5

Lycka till!

Uppgift 1. Beräkning av uttryck (4p)

Vi ger DrRacket följande uttryck att beräkna (i ordningen som anges).

Vilka värden får `a, b, c, ...`? Om det uppstår fel vid definitionerna, beskriv vad för fel det är (vilken typ).

```
> (define a 'haha)
> a
> (define b (+ a 1))
> b
> (define (p p) (p))
> p
> (define c (p (lambda () 3)))
> c
> (define d (cons a m))
> d
> (define (e x) (* x 2))
> e
> (define h (lambda (g) (e)))
> h
> (define i (h 3))
> i
```

Uppgift 2. Datatyp: länkad lista (4p)

a) Antag att vi har evaluerat följande Racket-uttryck (i ordning).

```
(define x (list 2 3))
(define y (cons 1 x))
(define z (cons (cons x y) 99))
(define w (list 4 (list 5 6)))
```

Rita box-pointer diagram för `x, y, z, w`. Var noga med pekarna! (2p)

b) Vi evaluerar följande uttryck:

```
(define p (mcons 1 2))
(define q (mcons 3 4))
(set-mcdr! p q)
(set-mcdr! p 999)
```

Går det att skriva uttryck utan `set-mcar!` och `set-mcdr!`, som genererar samma struktur? Skriv kod isåfall. Motivera annars kortfattat (max två meningar) varför det inte går. (1p)

c) Vi evaluerar följande uttryck:

```
(define r (mcons 5 6))
(define s (mcons 7 8))
(set-mcdr! r s)
(set-mcdr! r r)
```

Går det att skriva uttryck utan `set-mcar!` och `set-mcdr!`, som genererar samma struktur? Skriv kod isåfall. Motivera annars kortfattat (max två meningar) varför det inte går. (1p)

Uppgift 3. Tillstånd och omgivningsdiagram (5p)

Vi ges följande kod:

```
(define x 100)

(define (f x)
  (define (g y)
    (+ x y))
  (g x))

(f 12)
```

a) När vi evaluerar koden returneras ett värde. Vilket? (1.5p)

b) På nästa sida ser du fyra omgivningsdiagram. Vilket av dem visar det som händer när vi evaluerar koden ovan? Enbart en bokstav (A/B/C/D) behövs.

Korrekt svar ger poäng, felaktigt ger 1p avdrag på uppgiftens poäng[1]. Du behöver inte svara om du inte vill. (1.5p)

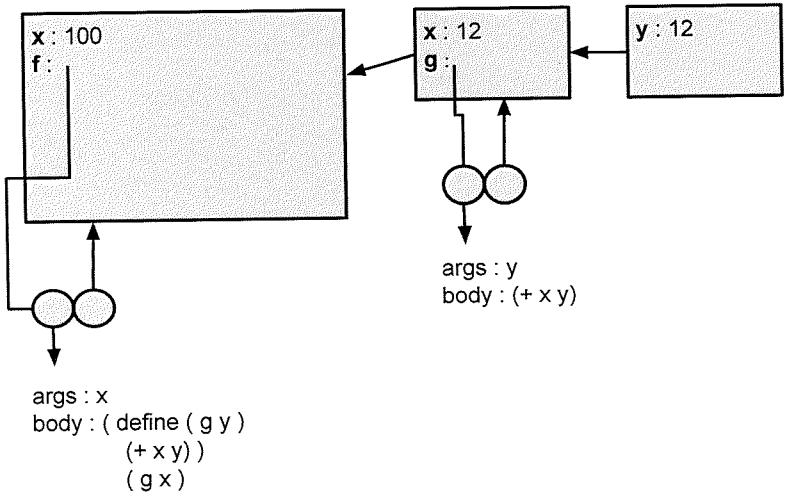
c) Vi evaluerar följande:

```
(define cat 'tesco)
(define (change-name! name)
  (set! name cat)
  (set! name 'forex))
(change-name! 'tiger)
```

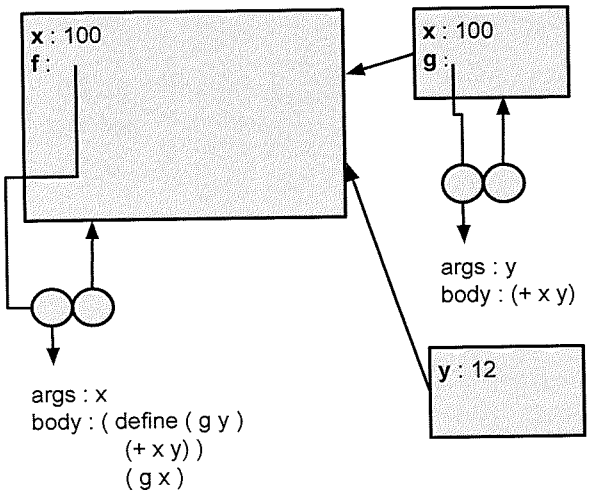
Vilket värde har `cat` efteråt? Motivera kortfattat (en-två meningar). (2p)

[1]Uppgift 3 kan alltså inte ge minuspoäng.

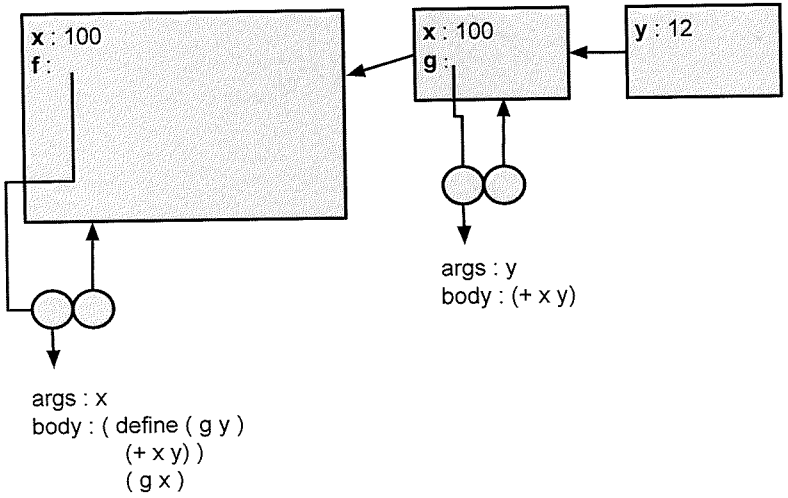
A


x : 100
f :
x : 12
g :
y : 12
args : y
body : (+ x y)
args : x
body : (define (g y)
(+ x y))
(g x)

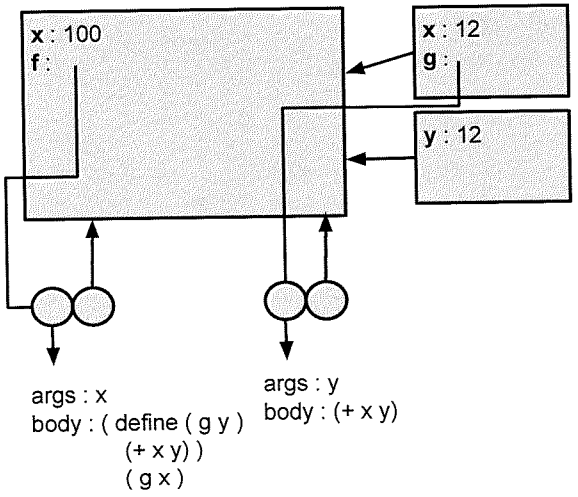
B


x : 100
f :
x : 100
g :
args : y
body : (+ x y)
y : 12
args : x
body : (define (g y)
(+ x y))
(g x)

C


x : 100
f :
x : 100
g :
y : 12
args : y
body : (+ x y)
args : x
body : (define (g y)
(+ x y))
(g x)

D


x : 100
f :
x : 12
g :
y : 12
args : x
body : (define (g y)
(+ x y))
(g x)
args : y
body : (+ x y)

Uppgift 4. Enkelrekursion över olika domäner (4p)

a) Konstruera funktionen `sum-even` som tar en lista av tal, och returnerar summan av de jämna talen i listan. Så här ska den fungera:

```
> (sum-even '(1 2 3 4 5))
6
> (sum-even '(100 1 -100))
0
> (sum-even '())
0
```

Ange kod för funktionen. Du får använda Racket-funktionen `even?`. (2p)

b) Antag att du får använda Racket-funktionen `expt` som motsvarar upphöjt-till. x^y blir `(expt x y)`. Med Knuths dubbelpilsnotation kan vi skriva $a \uparrow\uparrow 2 = a^a$, $a \uparrow\uparrow 3 = a^{(a^a)}$, $a \uparrow\uparrow 4 = a^{(a^{(a^a)})}$ och så vidare. Lite informellt, bygger vi ett torn av exponenter.

Definiera `(tower a n)` så att `(tower a n)`$=a \uparrow\uparrow n$ för godtyckligt a, och heltal $n \geq 1$. Så här ska det fungera:

```
> (tower 2 1)    ;; 2^1
2
> (tower 2 3)    ;; 2^(2^2) = 2^4 = 16
16
> (tower 4 2)    ;; 4^4 = 256
256
> (tower 4 3) ;; 4^(4^4) = 4^16
134078079299425970995740249982058461274793658205923933
777235614437217640300735469768018742981669034276900318
581864860508537538828119465699464336490060840 96
```

Svara med kod. (2p)

Uppgift 5. Abstrakt datatyp, högre ordningens procedur (5p)

OBS! Försök gärna besvara alla deluppgifter, även om du inte gjort 5a!

Uppgift 5a) Representation (2p)

I matematiken stöter vi på polynom, som exempelvis $p(x) = 8x^3 + 7x^2 + 6x + 5$. Vi vill nu skapa en abstrakt datatyp `poly` som representerar ett polynom. När vi skapar polynomet anger vi helt enkelt dess lista av koefficienter, där vi börjar med konstanttermen. p ovan skulle alltså ha koefficient-listan (5 6 7 8), och polynomet $q(x) = x^4 + 3x^2 + 7$ skulle ha listan (7 0 3 0 1). Det sista ser vi kanske tydligare om vi skriver ut det som $q(x) = 1x^4 + 0x^3 + 3x^2 + 0x^1 + 7x^0$. Ett polynoms grad (en: degree) är den högsta förekommande exponenten. $p(x)$ har grad 3, $r(x) = 5$ har grad 0.

Din uppgift är att implementera `poly`-typen, och dessa fyra funktioner:

- En konstruktor `make-poly` som tar en lista av koefficienter och skapar ett polynom. Vi kan anta att användaren anger polynom utan avslutande nollor (så $p(x)$ ovan skapas alltid med listan (5 6 7 8), och inte t ex (5 6 7 8 0 0 0...)).
- Ett predikat `poly?` som kontrollerar om ett objekt är ett polynom[2].
- `get-degree` tar ett polynom och returnerar dess grad.
- `get-coeff` tar ett polynom och en exponent k ($k \geq 0$ heltal), och returnerar koefficienten framför termen x^k.

Så här ska det fungera:

```
> (define p (make-poly '(5 6 7 8)))
> (poly? '(5 6 7 8))     ;; var listan ett polynom?
#f
> (poly? p)      ;; blir det vi skapade ett polynom?
#t
> (get-degree p)
3
> (get-coeff p 3)       ;; ge mig koefficienten vid x^3
8
> (get-coeff p 5)       ;; ge mig koefficienten vid x^5
0
```

OBS! Du kan ha hjälp av funktionen `list-ref`, som givet en lista och ett index i (0,...) returnerar elementet på plats i. `(list-ref 1 '(a b c))` => `b`.

[2]Korrekt utdata från make-poly. Du behöver inte kontrollera att användaren gjort rätt när de anropade make-poly.

Uppgift 5b) Användning (3p)

Polynom av en variabel kan såklart adderas, och bildar då nya polynom. Om vi till exempel lägger ihop $p(x) = 8x^3 + 7x^2 + 6x + 5$ och $q(x) = x + 95$, får vi det nya polynomet $8x^3 + 7x^2x + (6 + 1)x + (5 + 95) = 8x^3 + 7x^2 + 7x + 100$.

Mer allmänt, skrivs polynom som summor av termer på formen $c_k * x^k$. När vi adderar två polynom $p(x), q(x)$ går vi igenom dem term för term, och låter koefficienten framför x^k i det nybildade polynomet bli summan av koefficienten framför x^k i $p(x)$ och koefficienten framför x^k i $q(x)$.[3]

Din uppgift är att skriva en funktion `add-poly` som tar två polynom och returnerar ett nytt polynom som är summan av polynomen. Tänk på att du inte får anta något mer om polynomet än att funktionerna ovan (i 5a-beskrivningen ovan) kan användas. Så här ska det fungera:

```
> (define p (make-poly '(5 6 7 8))) ;; p som ovan
> (define q (make-poly '(95 1))) ;; q(x) = x + 95
> (define h (add-poly p q));; h(x) = 8x^3 + 7x^2 + 7x + 100
> (get-degree h)
3
```

För full poäng, notera särskilt:

```
> (define r (make-poly '(1 2 -7 -8)))
> (define s (add-poly p r)) ;; s(x) = 8x + 6
> (get-degree s)
1
```

[3]Om polynomen har olika grad, antar vi att koefficienter som "saknas" är 0.

Uppgift 6. Högre ordningens procedur (5p)

a) Vi säger att $h = f \circ g$, sammansättningen av f och g, om $h(x) = f(g(x))$. Skriv en funktion `compose` som tar två funktioner och returnerar sammansättningen. Du kan anta att båda funktionerna tar ett argument. Så här ska det fungera:

```
> (define square (lambda (x) (* x x)))
> (define double (lambda (x) (* 2 x)))
> (define h1 (compose square double))
> (h1 3)   ;; h1(3) = square(double(3)) = square(6) = 36
36
```

Redovisa kod för funktionen. (2p)

b) Om vi har en funktion f säger vi att $f^1(x) = f(x), f^2(x) = f(f(x)), f^3(x) = f(f(f(x)))$ och så vidare. Skriv en funktion `repeat` som tar en funktion f och ett antal repetitioner $n \geq 1$, och returnerar funktionen f^n. Din lösning *måste* använda `compose`.[4].

Så här ska det fungera:

```
> (define sq2 (repeat square 2))
> (define sq3 (repeat square 3))
> (sq2 3) ;; (3^2)^2 = 9^2 = 81
81
> (sq3 3) ;; ((3^2)^2)^2 = (9^2)^2 = 81^2
6561
> (define dbl2 (repeat double 2))
> (dbl2 7) ;; double(double(7)) = double(14) = 28
28
```

Redovisa kod för funktionen. `repeat` ska givetvis kunna ta valfri funktion (som tar ett argument) som indata, och får inte bara fungera för square. (2p)

c) Ovan har du $n = 1$ som basfall. Antag att du ville ha $n = 0$ som basfall istället. Vad för funktion borde `(repeat f 0)` returnera, om lösningen ska vara konsekvent? (1p)

[4]Vi kommer att anta att compose fungerar som den ska i 6a. Du kan få full poäng på deluppgiften även om du inte besvarat, eller gjort fel i, 6a.