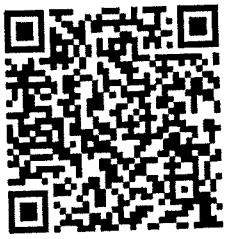
Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-02-08
Sal (2)	TER1 TERE
Tid	8-10
Kurskod	TDDC74
Provkod	KTR1
Kursnamn/benämning **Provnamn/benämning**	Programmering - abstraktion och modellering Frivillig dugga
Institution	IDA
Antal uppgifter som ingår i tentamen	5
Jour/Kursansvarig Ange vem som besöker salen	Jalal Maleki
Telefon under skrivtiden	ankn. 1963
Besöker salen ca klockan	ja
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund anna.grabska.eklund@liu.se ankn. 2362
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet



Datum för tentamen	2016-02-08
Sal (2)	TER1 TERE
Tid	8-10
Kurskod	TDDC74
Provkod	KTR1
Kursnamn/benämning **Provnamn/benämning**	Programmering - abstraktion och modellering Frivillig dugga
Institution	IDA
Antal uppgifter som ingår i tentamen	5
Jour/Kursansvarig Ange vem som besöker salen	Jalal Maleki
Telefon under skrivtiden	ankn. 1963
Besöker salen ca klockan	ja
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund anna.grabska.eklund@liu.se ankn. 2362
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

TDDC74 Programmering: Abstraktion och modellering
Dugga 1, kl 8—10, 8 feb 2016

Läs alla frågorna först och bestäm dig för i vilken ordning du vill lösa uppgifterna. Uppgifterna är inte ornade 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.

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! Du får använda `define` för att definiera procedurer och namn. Men du får inte använda det (eller `set!`) för att ändra på/uppdatera värden associerade med namn. Tanken är att lösningarna ska vara strikt funktionella.

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

Lycka till!

Uppgift 1 (3 poäng)

Antag att vi har evaluerat följande uttryck i Scheme:

```
(define f
  (lambda (n) (* n 7)))
(define g
  (lambda (m)
    (lambda (n) (- n m))))
(define x 5)
(define y (if (< x x) x (* -1 x)))
```

Vad blir värdet av följande uttryck? Om evaluering av uttrycket leder till fel, förklara varför felet uppstår.

```
1. (+ x y)
2. (f x)
3. (g x)
4. (f (g y))
5. (g x y)
6. ((g x) y)
```

Uppgift 2 (2 poäng)

Omskrivningar:

i) Skriv om `cond`-uttrycket i följande definition som `if`-uttryck. Var noga med indentering (indrag)!

```
(define ok-num?
  (lambda (n)
    (cond
      ((< n 100) #t)
      ((even? n) #f)
      (else #t))))
```

ii) Skriv om definitionen nedan genom att ersätta `let` med dess motsvarande `lambda`-uttryck.

```
(define larger-sqr
  (lambda (m n)
    (let ((m2 (* m m))
          (n2 (* n n)))
      (if (> m2 n2) m2 n2))))
```

Uppgift 3 (3 poäng)

a) Nedan ges vi fyra korrekta funktioner. Klassificera processerna de ger upphov till. Alternativen är *linjärrekursiv*, *trädrekursiv* och *iterativ process*.
Du behöver inte motivera dina svar men svara bara om du är säker. Inkorrekta svar ger poängavdrag. Deluppgiften ger som minst 0p.[1] Du behöver inte svara på alla, om du inte vill.

i)
```
(define f
  (lambda (k res)
    (if (= k 0)
        res
        (+ (* 2 k)
           (f (- k 1) (+ res 1))))))
```

ii)
```
(define g
  (lambda (m)
    (if (= m 0)
        m
        (g (- m 1)))))
```

iii)
```
(define h
  (lambda (n)
    (if (<= n 0)
        1
        (* n (h (- n 2))))))
```

iv)
```
(define fn
  (lambda (n)
    (if (< n 2)
        1
        (+ (fn (- n 1))
           (fn (- n 2))))))
```

b) Motivera *ett* av dina val (i-iv) kortfattat. *Varför* är det den typ av process du beskriver?

[1] D v s du kan inte få minuspoäng på deluppgiften.

Uppgift 4 (2 poäng)

Ett positivt heltal $n > 1$ är ett *primtal* om det enbart är delbart med 1 och n. Talet 17 är enbart delbart med 1 och 17, och är därför ett primtal. Talet 12 är delbart med 1, 2, 3, 4, 6, och 12, och är därmed inget primtal. Talet 9 är inte heller ett primtal, eftersom det är delbart med 1, 3, och 9.

Skriv ett predikat `prime?` som tar ett heltal $n > 1$ och returnerar #t om n är ett primtal, och #f annars. Exempel på anrop:

```
> (prime? 5)
#t
> (prime? 11)
#t
> (prime? 9)          ;; 9 är delbart med 3 och därmed inte primtal
#f
```

Uppgift 5 (2 poäng)

Skriv funktionen `twice` som tar en funktion `f` som argument och returnerar en funktion som beräknar `(f (f x))` vid korrekt anrop. Som framgår förväntas `f` vara en funktion me 1 parameter. Exemplen nedan visar hur `twice` fungerar.

```
> (define add3 (lambda (x) (+ x 3)))

> ((twice add3) 5)
11

> ((twice sqr) 2)                    ;sqr tar ett tal och ger kvadraten
16

> ((twice (twice add3)) 5)
17

> ((twice (lambda (x) (* x 3))) 4)
36
```