TÖL304G Miðmisserispróf
TÖL304G Midterm Exam

1. Engin hjálpargögn eru leyfileg. No help materials are allowed.
2. Skrifið svörin á þessar síður. Write your answers on these pages.
3. Ef svarið kemst ekki fyrir á svæðinu sem ætlað er fyrir það má halda áfram að skrifa á auðu síðurnar aftast, en þá skal segja frá því í því svæði sem ætlað er fyrir svarið, til dæmis „sjá framhald á síðu 11“.
   If the answer does not fit on the allotted space you may continue your answer on the empty pages at the end, but in that case you should write an indication to that effect on the space allotted for the answer, for example “continued on page 11”.
4. Forðist að krumpa eða rífa þessar blaðsíður, þær þurfa að fara gegnum skanna. **Skrifið skýrt með dökku letri og ekki skrifa í blájaðrana.**
   Refrain from mutilating or tearing these pages, they have to go through a scanner. **Write clearly with dark letters and do not write in the extreme margins.**
5. Baksíður blaðsíðanna verða ekki skannaðar og má nota fyrir krass. The backs of the pages will not be scanned and may be used for scratch.

Nafn/Name: ……..

Háskólatölvupóstfang/University Email: ……..

1. **(5%)** Sýnið BNF, EBNF eða málrit fyrir mál segða yfir stafrófið $\{a, +, (,)\}$. Svigar verða að vera í jafnvægi, $+$ er tvíundaraðgerð og a er breytunafn, sem er eina leyfða frumstæða segðin.
   Show BNF, EBNF or syntax diagrams for the language of expressions over the alphabet $\{a, +, (,)\}$. Parentheses must be balanced, $+$ is a binary operation, and a is a variable name, which is the only allowed primitive expression.

   Dæmi um strengi í málinu. Examples of strings in the language.

   a
   $a + a + a$
   (a)
   $(((a)))$
   $(a) + (a)$
   $a + (a + a + (a + a) + a) + a + a$

   Dæmi um strengi ekki í málinu. Examples of strings not in the language.

   ϵ (tómi strengurinn, the empty string)
   $($
   $)$
   $+a$
   $((a)$
   aa
   b

   Svar:
   EBNF:
   e = ‘(‘, e, ‘)‘ | e, ‘+‘, e | ‘a‘ ;
   BNF:
   <e> ::= (<e>) | <e>+<e> | a

2. (10%)

a. Íhugið eftirfarandi EBNF skilgreiningu fyrir mál yfir stafrófið $\{(,),a\}$.
Consider the following EBNF definition for a language over the alphabet $\{(,),a\}$.

e = ‘a’ | e, ‘(‘, e, ‘)’ ;

Hverjir eftirfarandi strengja eru í málinu sem skilgreint er? Tvö röng svör gefa núll stig.
Which of the following strings are in the language defined? Two wrong answers give zero points.

1. a(a(a))
2. a (a)(a)
3. a
4. (
5.)
6. (a)
7. a(a)
8. a)a(
9. a()
10. a(a)()

Svar:

1	2	3	4	5	6	7	8	9	10
x	x	x				x			

b. Hverjar af eftirfarandi reglulegu segðum skilgreina eitthvert mál yfir stafrófið $\{a, b\}$ þar sem a og b eru í jafnvægi, þ.e. strengi þannig að ef öllum $'a'$ væri breytt í $'('$ og öllum $'b'$ breytt í $')'$ þá væru svigar í jafnvægi? Tvö röng svör gefa núll stig.
Which if the following regular expressions define some language over the alphabet $\{a, b\}$ where a and b are balanced, i.e. strings such that if all $'a'$ were to be changed to $'('$ and all $'b'$ changed to $')'$ then parentheses would be balanced? Two wrong answers give zero points.

1. a^*b^*
2. $(a(a(ab)^*b)^*b)^*$
3. $aa(aabb)^*bbab$
4. $(ab|a|b)^*$
5. $(ab)^*$
6. $a(bbaa)^*b$
7. $a(ba)^*b$
8. $(ab|a(ab)^*b)^*$

Svar:

1	2	3	4	5	6	7	8
	x	x		x		x	x

3. (15%)

Hverjar eftirfarandi fullyrðinga um lokanir eru sannar? Tvö röng svör gefa núll punkta.

Which of the following statements about closures are true? Two wrong answers give zero points.

a) Lokanir eru til í C. Closures exist in C.
b) Lokanir eru til í Scheme. Closures exist in Scheme.
c) Lokanir eru til í CAML. Closures exist in CAML.
d) Lokanir eru til í Morpho. Closures exist in Morpho.
e) Lokanir eru nauðsynlegar til að skila staðværu falli sem gildi í bálkmótuðum forritunarmálum. Closures are necessary in order to return a local function as a value in block structured programming languages.
f) Lokanir innihalda straum. Closures contain a stream.
g) Lokanir innihalda fallsbendi. Closures contain a function pointer.
h) Lokanir má nota til að útfæra strauma í Scheme. Closures can be used to implement streams in Scheme.
i) Lokanir innihalda tengihlekk (aðgangshlekk). Closures contain an access link (static link).
j) Lokanir innihalda stýrihlekk. Closures contain a control link (dynamic link).
k) Lokanir eru aðeins mögulegar ef vakningarfærslur eru í kös. Closures are only possible if activation records are in the heap.

Svar:

a	b	c	d	e	f	g	h	i	j	k
	x	x	x	x		x	x	x		

4. **(15%)** Skrifið **halaendurkvæmt** fall í Scheme, CAML eða Morpho sem (í Morpho) hefur eftirfarandi lýsingu. Ekki má nota lykkju og ekki má gefa breytu nýtt gildi. Nota má hjálparföll sem þið þurfið þá að skrifa sjálf **ásamt lýsingu þeirra**. Aðeins má nota einfaldar innbyggðar aðgerðir í Scheme, CAML og Morpho, þ.e. aðgerðir sem hafa tímaflækju O(1) svo sem cons, car, cdr og null?.
   Write a **tail recursive** function in Scheme, CAML or Morpho, which (in Morpho) has the following description. No loops are allowed and it is not allowed to give a variable a new value. Helper functions can be used, which you must then write yourself, **along with their descriptions**. Only simple built-in operations in Scheme, CAML and Morpho may be used, i.e. operations that have time complexity O(1) such as cons, car, cdr and null?.

```
;;; Notkun: x = iota(n);
;;; Fyrir:  n er heiltala, n>=0.
;;; Gildi:  x er listinn [1,2,…,n]

;;; Usage:  x = iota(n);
;;; Pre:    n is an integer, n>=0.
;;; Value:  x is the list [1,2,…,n]
```

   Svar:
   Morpho:

```
rec fun iota(n)
{
    ;;; Notkun: x = help(i,z);
    ;;; Fyrir:  0 <= i <= n,
    ;;;          z er listinn [i+1,i+2,…,n]
    ;;; Gildi:  Listinn [1,2,…,n]
    rec fun help(i,z)
    {
        if( i==0 ) { return z };
        help(i-1,i:z);
    };
    help(n,[]);
};
```

Scheme:

```
(define (iota n)
    ;; Notkun: (help i z)
    ;; Fyrir:  0 <= i <= n,
    ;;         z er listinn (i+1 i+2 … n)
    ;; Gildi:  Listinn (1 2 … n)
    (define (help i z)
        (if (= i 0)
            z
            (help (- i 1) (cons i z))
        )
    )
    (help n '())
)
```

5. (15%)

Skrifið eftirfarandi fall í Scheme, CAML eða Morpho. Notið aðeins einfaldar aðgerðir og engar lykkjur eins og í dæminu á undan. Write the following function in Scheme, CAML or Morpho. Use only simple operations and no loops as in the previous question.

```
;; Notkun: (foldl f u x)
;; Fyrir:  f er tvíundarfall,
;;         u er eitthvert gildi,
;;         x=(x1 … xN) er listi einhverra gilda
;; Gildi:  (f (f … (f (f u x1) x2) …) xN)
;; Ath.:   (foldl f u '()) skal skila u

;; Usage:  (foldl f x u)
;; Pre:    f is a binary function,
;;         u is some value,
;;         x=(x1 … xN) is a list of some values
;; Value:  (f (f … (f (f u x1) x2) …) xN)
;; Note:   (foldl f u '()) should return u
```

Svar:

```
(define (foldl f u x)
    (if (null? x)
        u
        (foldl f (f u (car x)) (cdr x))
    )
)
```

6. (15%)

Skrifið eftirfarandi powerList fall í Scheme, CAML eða Morpho. Lýsingin í Scheme er eftirfarandi. Þið megið reikna með því að til sé fall append sem skeytir saman tveimur listum ásamt map falli fyrir lista með venjulegri lýsingu. Í Morpho má nota lykkjur, en þá þurfa þær að hafa fastayrðingar lykkju.

Write the following powerList function in Scheme, CAML or Morpho. The description in Scheme is the following. You may assume the existence of a function append that concatenates two lists as well as a map function for lists with the usual description. In Morpho you may use loops, but then they must have loop invariants.

```
;; Notkun: (powerList n)
;; Fyrir:  n er heiltala, n>=0.
;; Gildi:  Listi allra undirlista listans
;;         (n … 2 1).
;;         Þetta er listi sem inniheldur 2^n
;;         undirlista.

;; Usage:  (powerList n)
;; Pre:    n is an integer, n>=0.
;; Value:  The list of all sublists of the list
;;         (n … 2 1).
;;         This is a list that contains 2^n
;;         sublists.
```

Svar:

Scheme:

```
(define (powerList n)
  (if (= n 0)
      '(())
      (append
         (powerList (- n 1))
         (map
            (lambda (z) (cons n z))
            (powerList (- n 1))
         )
      )
  )
)
```

```
Morpho:
  rec fun powerList(n)
  {
    var pl = [[]];
    var i = 0;
    while( i!=n )
      ;;; 0 <= i <= n.
      ;;; pl er listi allra undirlista listans
      ;;; [i,i-1,…,2,1].
    {
      i = i+1;
      pl = append(pl,map(fun(z){i:z},pl));
    };
    pl
  };
```

7. (15%)

Skrifið fall cross í Scheme eða Morpho sem hefur eftirfarandi lýsingu í Scheme.

Write a function cross in Scheme or Morpho that has the following description in Scheme.

```
;; Notkun: (cross a b)
;; Fyrir:  a og b eru óendanlegir straumar,
;;         a=[a1 a2 a3 ...]
;;         b=[b1 b2 b3 ...]
;; Gildi:  Óendanlegi straumurinn af
;;         óendanlegum straumum af tveggja
;;         staka listum
;;           [[(a1 b1) (a1 b2) (a1 b3) ...]
;;            [(a2 b1) (a2 b2) (a2 b3) ...]
;;            [(a3 b1) (a3 b2) (a3 b3) ...]
;;            .
;;            .
;;            .
;;           ]
;;         sem inniheldur nokkurs konar
;;         krossmargfeldi a og b.

;; Usage:  (cross a b)
;; Pre:    a and b are infinite streams,
;;         a=[a1 a2 a3 ...]
;;         b=[b1 b2 b3 ...]
;; Value:  The infinite stream of infinite
;;         streams of two element lists
;;           [[(a1 b1) (a1 b2) (a1 b3) ...]
;;            [(a2 b1) (a2 b2) (a2 b3) ...]
;;            [(a3 b1) (a3 b2) (a3 b3) ...]
;;            .
;;            .
;;            .
;;           ]
;;         that contains a sort of cross
;;         product of a and b.
```

Vísbending: Í Scheme megið þið reikna með að allar okkar straumaaðgerðir séu til staðar og í Morpho megið þið reikna með að til séu fall streamMap sem virkar svipað og samsvarandi straumafall í Scheme. Athugið að til að senda fall f sem viðfang í Morpho þarf að senda lokun svo sem `fun(x){f(x)}`.
Hint: In Scheme you may assume the existence of all our stream operations and in Morpho you may assume the existence of a function streamMap that works similarly to the corresponding Scheme function. Note that to send a function f as an argument in Morpho you need to send a closure such as `fun(x){f(x)}`.

Svar:
Scheme:

```
(define (cross a b)
   (cons-stream
      (stream-map
         (lambda (z) (list (stream-car a) z))
         b
      )
      (cross (stream-cdr a) b)
   )
)
```

Morpho:

```
rec fun cross(a,b)
{
  val ha=streamHead(a), ta=streamTail(a);
  #[streamMap(fun(z){[ha,z]},b)$cross(ta,b)];
};
```

8. (10%)

Vakningarfærsla falls í bálkmótuðu forritunarmáli eins og Scheme inniheldur sum eftirfarandi atriða. Hver? Eitt rangt svar gefur núll stig.
The activation record (stack frame) of a function in a block-structured programming language such as Scheme contains some of the following. Which? One wrong answer gives zero points.

a) Vendivistfang. A return address.
b) Benda á öll föll sem hægt er að kalla á úr fallinu. Pointers to all the functions that can be called from the function.
c) Alla hluti sem til eru í kerfinu. All objects that exist in the system.
d) Vakningarfærslur allra falla sem hægt er að kalla á. The activation records of all functions that can be called.
e) Skráakerfi tölvunnar. The file system of the computer.
f) Staðværar breytur fallsins sem kallaði á fallið. The local variables of the function that called the function.
g) Nöfn allra falla sem hægt er að kalla á. The names of all functions that can be called.
h) Benda á allar lifandi vakningarfærslur. Pointers to all living activation records.
i) Staðværar breytur fallsins. The local variables of the function.
j) Víðværar breytur sem eru aðgengilegar í fallinu. The global variables that are accessible in the function.
k) Viðföng fallsins. The arguments of the function.
l) Aðgangshlekk (tengihlekk). An access link (static link).
m) Stýrihlekk. A control link (dynamic link).

Svar:

a	b	c	d	e	f	g	h	i	j	k	l	m
x								x		x	x	x

(Aukablaðsíða / Extra page)

(Aukablaðsíða / Extra page)