TÖL304G Forritunarmál
Miðannarpróf – Lausn
TÖL304G Programming Languages
Midterm Exam – Solution

Leysið 5 af eftirfarandi dæmum og a.m.k. eitt úr hverjum kafla. Ef þið leysið fleiri en 5 dæmi skuluð þið tilgreina hvaða dæmi eiga að gilda til einkunnar, að öðrum kosti verður tekið meðaltal allra þeirra sem þið skilið. Öll dæmi gilda jafnt.

Solve 5 of the following exercises and at least one from each chapter. If you solve more than 5 then you sould specify which solutions should count toward the grade, otherwise the average of the grades for the solved exercises will be taken. All exercises count equally.

Engin hjálpargögn eru leyfileg. Prófið gildir sem ein dæmaskil. Skrifið nafn ykkar og upphafsstafi dæmakennara ykkar (GBB, SA eða ÍHS[1]) á úrlausnarblöðin.

No help materials are allowed. The exam counts as one home assignment. Write your name and the initials of your tutor (GBB, SA or ÍHS) on the solution papers.

Kafli I - Almennt efni

Chapter I - General problems

1. Sýnið BNF, EBNF eða málrit fyrir mál λ-reiknisegða með svigum, breytunöfnunum x, y og z, fallsbeitingu NM þar sem N og M eru λ-reiknisegðir, og fallsskilgreiningum $\lambda a.N$ þar sem a er breytunafn og N er λ-reiknisegð. Dæmi um löglegar formúlur eru þá t.d. $\lambda x.xy$, $x(xy)$, xyz og $(\lambda x.yx)z$, en ekki t.d. a eða $\lambda xy.z$.

   Show BNF, EBNF or syntax diagrams for a language og λ-expressions with parentheses, the variable names x, y and z, function applications NM where N and M are λ-expressions, and function definitions $\lambda a.N$ where a is a variable name and N is a λ-expression. Examples of legal expressions are $\lambda x.xy$, $x(xy)$, xyz and $(\lambda x.yx)z$, but not, for example a or $\lambda xy.z$.

   Lausn: BNF:

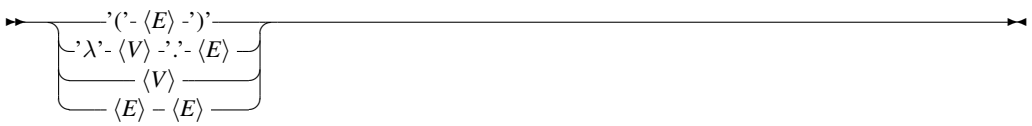
   <E> ::= (<E>) | λ <V> . <E> | <V> | <E> <E>
   <V> ::= x | y | z

   EBNF:

   e = '(', e, ')' | 'λ', v, '.', e | v | e, e ;
   v = 'x' | 'y' | 'z' ;

   Málrit:

   <E>:

   <V>:

[1] GBB kennir kl. 16:40 á mánudögum, SA kl. 8:20 á þriðjudögum og ÍHS kl. 16:40 á þriðjudögum.

2. Íhugið málin sem skilgreind eru annars vegar með reglulegu segðinni $(a(a(a(ab)^*b)^*b)^*b)^*$ og hins vegar með BNF mállýsingunni (athugið að ϵ stendur fyrir tóma strenginn) að neðan.

   S ::= a <S> b <S> | ϵ

   Consider the languages defines on one hand by the regular expression $(a(a(a(ab)^*b)^*b)^*b)^*$ and on the other hand by the BNF grammar above. Note that ϵ stands for the empty string.

   (a) Segið til um hvort reglulega segðin og BNF mállýsingin lýsa sama máli.
   Specify whether the regular expression and the BNF grammar describe the same language.
   Lausn: Nei, þetta eru tvö mismunandi mál.

   (b) Ef málin eru ekki þau sömu, lýsið þá báðum málunum í stuttum texta og sýnið streng sem er í öðru málinu en ekki hinu.
   If the languages are not the same then describe both languages in a short text and show a string that is in one of the languages but not the other.
   Lausn: Reglulega segðin lýsir málinu yfir stafrófið $\{a,b\}$ þar a verkar svipað og svigi opnast og b sem svigi lokast og svigar eru í jafnvægi og í mesta lagi fjórir svigar (a) eru opnir á hverjum stað í strengnum. BNF mállýsingin lýsir hins vegar málinu þar sem svigar eru í jafnvægi og engin takmörk eru á dýpt sviga. Strengurinn $aaaaabbbbb$ er í BNF málinu en ekki í hinu.

   (c) Ef málin eru þau sömu lýsið þá málinu sem þau lýsa í stuttum texta.
   If the languages are the same then describe the language that they describe in a short text.
   Lausn: Á ekki við.

3. Íhugið eftirfarandi forritstexta í einhverju ímynduðu forritunarmáli. (Spyrjið ef ykkur finnst merking eða málfræði ekki augljós.)

```
procedure f(x,y)
{
  x = 2;
  print x,y;
  y = 1;
}

int i,a[100];
for( i=0 ; i!=100 ; i++ ) a[i]=2*i+1;
f(a[0],a[a[0]]);
print a[0], a[1];
```

   Consider the above code in some imaginary programming language. (Ask if you feel the syntax or semantics is not obvious.)

   Hvað skrifar þetta forrit (fjögur gildi í hvert skipti) ef viðföngin eru: — What does this program write (four values each time) if the arguments are:

   - gildisviðföng — call by value?
     Lausn:

     2 3
     1 3

   - tilvísunarviðföng — call by reference?
     Lausn:

     2 3
     2 1

   - afritsviðföng — call by value-result?
     Lausn:

     2 3
     2 1

eða hugsanlega má skilja afritunina þannig að afritað sé til baka í `a[2]` (dálítið langsótt), en þá fæst niðurstaðan

2 3
2 3

- nafnviðföng — call by name?
  Lausn:

  2 5
  2 3

Kafli II - Bálkmótun

Chapter II - Block Structure

4. Gerið ráð fyrir földun í bálkmótuðu forritunarmáli eins og myndin að neðan sýnir. Forritstextarnir að neðan sýna dæmi um slíka földun í Scheme og Morpho. Földunin er þannig að F inniheldur G og I, og G inniheldur H.

   Gerið grein fyrir hvar í földuninni er hægt að kalla á hvert um sig af F, G, H og I. Gerið einnig grein fyrir hvar í földuninni er unnt að nota innri breytur í F, G, H og I.

   Földunin táknuð með innfellingu — The nesting shown by indentation:

```
F
  G
    H
  I
```

   Forritstexti í Scheme með slíka földun — Scheme code with the nesting:

```
(define (F ...)
  (define (G ...)
    (define (H ...)
      ...
    )
    ...
  )
  (define (I ...)
    ...
  )
  ...
)
```

   Forritstexti í Morpho með slíka földun — Morpho code with the nesting:

```
rec fun F(...)
    {
        rec fun G(...)
            {
                rec fun H(...)
                    {
                        ...
                    };
                ...
            },
            fun I(...)
            {
                ...
            };
        ...
    };
```

   Assume the nesting shown above in a block-structured programming language. The code snips above show examples of such nesting in Scheme and Morpho. In the nesting, F contains G and I, and G contains H.

   Specify where in the nesting it is possible to call each of F, G, H, and I. Also specify where in the nesting it is possible to use local variables of F, G, H, and I.
   Lausn:

   - Kalla má á F alls staðar í textanum, þ.e. í F, G, H og I.
   - Nota má staðværar breytur F alls staðar í textanum, þ.e. í F, G, H og I.
   - Kalla má á G alls staðar í textanum, þ.e. í F, G, H og I.
   - Nota má staðværar breytur G inni í G og H.
   - Kalla má á H inni í G og H.

- Nota má staðværar breytur H aðeins inni í H.
- Kalla má á I alls staðar, þ.e. í F, G, H og I.
- Nota má staðværar breytur I aðeins inni í I.

5. Íhugið eftirfarandi atriði og svarið spurningum um þau. Spurningarnar eru:

   (a) Er þetta atriði hluti af vakningarfærslum? Ef svo er, er það einungis í bálkmótuðum forritunarmálum.

   (b) Inniheldur þetta atriði bendi á vakningarfærslu? Ef atriðið bendir á vakningarfærslu, hver er þá tilgangurinn og er það einungis í bálkmótuðum forritunarmálum?

   Atriðin eru:

   (a) Staðværar breytur, svo sem breytur skilgreindar inni í falli — local variables, such as variables defined inside a function.
   Lausn: Þær eru hluti af vakningarfærslum, en innihalda að jafnaði ekki benda á vakningarfærslur.

   (b) Víðværar breytur, t.d. breytur utan allra falla í C++ — global variables, e.g. variables outside of all functions in C++.
   Lausn: Ekki hlutar af vakningarfærslum og innihalda að jafnaði ekki benda á vakningarfærslur.

   (c) Tilviksbreytur í hlutbundnum forritunarmálum — instance variables in object oriented programming languages.
   Lausn: Ekki hlutar af vakningarfærslum og innihalda að jafnaði ekki benda á vakningarfærslur.

   (d) Klasabreytur í hlutbundnum forritunarmálum — class variables in object oriented programming languages.
   Lausn: Ekki hlutar af vakningarfærslum og innihalda að jafnaði ekki benda á vakningarfærslur.

   (e) Viðföng falla — arguments of functions.
   Lausn: Þau eru hluti af vakningarfærslum, en innihalda að jafnaði ekki benda á vakningarfærslur.

   (f) Lokun — closure.
   Lausn: Ekki endilega í vakningarfærslum, en innihalda aðgangshlekk sem er bendir á vakningarfærslu. Tilgangurinn er að gefa aðgang að breytum sem eru í næstu földunarhæð fyrir ofan fallið sem lokunin stendur fyrir.

   (g) Stýrihlekkir — control links (dynamic links).
   Lausn: Nauðsynlegur hluti af vakningarfærlsum. Benda á vakningarfærslu þess falls sem snúa skal til baka til.

   (h) Vendivistföng falla — return addresses of functions.
   Lausn: Nauðsynlegur hluti af vakningarfærlum. Bendir á vélarmálsskipunina sem snúa skal til baka til, í fallinu sem kallaði.

   (i) Aðgangshlekkir (tengihlekkir) — access links (static links).
   Lausn: Nauðsynlegur hluti af vakningarfærslum í bálkmótuðum forritunarmálum en er ekki til staðar í öðrum forritunarmálum[2] Bendir á vakningarfærslu þá sem inniheldur breyturnar í næstu földunarhæð. Tilgangurinn er að gefa aðgang að breytum sem eru í næstu földunarhæð fyrir ofan núverandi fall.

   Consider the items listed above and answer the following questions for each item.

   (a) Is this item a part of activation records? If so, is it only in block-structured programming languages?

   (b) Does this item contain a pointer to an activation record? If so, what is the purpose and is it only in block-structured programming languages?

Kafli III - Fallsforritun og listavinnsla

6. Skrifið fall í Scheme, CAML eða Morpho sem tekur eitt viðfang sem er listi lista af fleytitölum og skilar summunni af margfeldinu af gildum innri listanna.

   Write a function in Scheme, CAML or Morpho that takes one argument that is a list of lists of floating point numbers and returns the sum of the products of the numbers in the inner lists.
   Lausn: (Í CAML Light)

[2] Aðgangshlekkurinn er það eina sem er í vakningarfærslum í bálkmótuðum málum en ekki í öðrum.

```
(*
Notkun: sumprod x
Fyrir:  x = [[x11;x12;...];...;[xK1;xK2;...]]
        er float list list (listi af listum
        af float).
Gildi:  (x11*x12*...)+...+(xK1*xK2*...)
*)
let sumprod x =
       it_list (prefix +.)
               0.0
               (map (function x->it_list (prefix *.) 1.0 x) x)
;;
```

Í Scheme (svipuð lýsing):

```
;; Notkun: (insert f u x)
;; Fyrir:  x=(x1 x2 ... xN) er listi.
;;         f er tvíundarfall.
;; Gildi:  (f (f ... (f (f u x1) x2) ...) xN)
(define (insert f u x)
  (if (null? x)
      u
      (insert f (f u (car x)) (cdr x))
  )
)

(define (sumprod x)
  (insert + 0.0 (map (lambda (x) (insert * 1.0 x)) x))
)
```

Eða, í Racket:

```
(define (sumprod x)
  (foldl + 0.0 (map (lambda (x) (foldl * 1.0 x)) x))
)
```

Eða í Morpho, frá grunni:

```
;;; Notkun: y = foldl(f,u,x);
;;; Fyrir:  f er tvíundaraðgerð, segjum f(x,y)=x!y,
;;;         u er gildi, x=[x1,...,xN] er listi.
;;; Eftir:  y = u!x1!x2!...!xN, reiknað frá vinstri.
rec fun foldl(f,u,x)
{
  x || (return u);
  foldl(f,f(u,head(x)),tail(x));
};

;;; Notkun: y = map(f,x);
;;; Fyrir:  f er einundarfall,
;;;         x=[x1,...,xN] er listi.
;;; Eftir:  y er listinn [f(x1),...,f(xN)].
rec fun map(f,x)
{
  x || (return []);
  f(head(x)) : map(f,tail(x));
```

```
};

;;; Notkun: y = sumprod(x);
;;; Fyrir:  x=[[x11,x12,...],[x21,x22,...],...]
;;;         er listi lista fleytitalna.
;;; Eftir:  y = (x11*x12*...)+(x21*x22*...)+...
rec fun sumprod(x)
{
  x = map(fun(x){foldl(fun(x,y){x*y;},1.0,x);},x);
  foldl(fun(x,y){x+y;},0.0,x);
};
```

7. Skrifið halaendurkvæmt[3] fall í Scheme, CAML eða Morpho, sem tekur eina heiltölu $n \geq 0$ sem viðfang og skilar listanum $[n, n-1, \ldots, 1]$.

   Write a tail recursive[4] function in Scheme, CAML or Morpho, that takes as argument one integer $n \geq 0$ and returns the list $[n, n-1, \ldots, 1]$.
   Lausn: CAML:

```
(*
Notkun: i n
Fyrir:  n er heiltala, n>=0
Gildi:  [n;n-1;...,1]
*)
let i n =
  (*
  Notkun: h k x
  Fyrir:  1<=k<=n+1, x=[x1;...;xN]
  Gildi:  [n;n-1;...;k;x1;...;xN]
  *)
  let rec h k x =
    if k>n then
      x
    else
      h (k+1) (k::x)
  in
    h 1 []
;;
```

   Í Scheme (sömu lýsingar):

```
(define (i n)
  (define (h k x)
    (if (> k n)
      x
      (h (+ k 1) (cons k x))
    )
  )
  (h 1 '())
)
```

   Í Morpho (sömu lýsingar):

[3] Það dugar að útfærslan noti halaendurkvæmt hjálparfall til að takmarka dýpt endurkvæmninnar. Það má ekki leysa þetta með lykkju í Morpho.

[4] It is enough that the implementation use a tail recursive help function to restrict the depth of the recursion. You may not do a loop implementation in Morpho.

```
rec fun i(n)
{
    rec fun h(k,x)
    {
        (k>n) && (return x);
        h(k+1,k:x);
    };
    h(1,[]);
};
```

8. Forritið straum í Scheme eða Morpho sem inniheldur óendanlegu rununa $1, 3, 5, 7, \ldots$ af öllum jákvæðum oddatölum. Ef ykkur vantar hjálparföll verðið þið að skilgreina þau.

   Program a stream in Scheme or Morpho that contains the infinite series $1, 3, 5, 7, \ldots$ of all positive odd numbers. If you need helper functions you need to program these as well.
   Lausn: Í Scheme:

```
;; Notkun: (map-stream f x)
;; Fyrir:  x=[x1 x2 ...] er óendanlegur straumur.
;;         f er einundarfall.
;; Gildi:  Óendanlegi straumurinn [(f x1) (f x2) ...]
(define (map-stream f x)
  (cons-stream (f (stream-head x)) (map-stream f (stream-tail x)))
)

;; Straumurinn [1 3 5 ...]
(define odds (cons-stream 1 (map-stream (lambda (x) (+ x 2)) odds)))
```

   Í Morpho:

```
;; Notkun: z = mapStream(f,x);
;; Fyrir:  x=#[x1,x2,...] er óendanlegur straumur.
;;         f er einundarfall.
;; Eftir:  z er óendanlegi straumurinn #[f(x1),f(x2),...]
rec fun mapStream(f,x)
{
    #[f(streamHead(x))$mapStream(f,streamTail(x))];
};

;;; Straumurinn #[1,3,5,...]
rec val odds = #[1$mapStream(fun(x){x+2;},odds)];
```

Kafli IV - Einingaforritun

Chapter IV - Modular Programming

9. Skrifið

   - hönnunarskjal,
   - fastayrðingu gagna
   - og smíð (forritun) á aðgerð til að bæta við gildi

   fyrir forgangsbiðraðaeiningu (priority queue) í Morpho. Athugið að biðröð og forgangsbiðröð eru ekki það sama.

   Athugið að aðeins þarf að forrita eina aðgerð. Athugið einnig að ekki er beðið um hraðvirka útfærslu.

   Munið upplýsingahuld.

   Write

- a design document,
- a data invariant,
- and the implementation (programming) of an operation to insert a value

for a priority queue module in Morpho. Note that a queue and a priority queue are not the same.

Note that you only need to implement one operation. Also note that your implementation need not be fast.

Remember information hiding.

Lausn:

```
;;; Útflutt
;;; =======
;;;   Notkun: p = makePQ();
;;;   Eftir:  p er ný tóm forgangsbiðröð
;;;
;;;   Notkun: add(p,x);
;;;   Fyrir:  p er forgangsbiðröð,
;;;           x er gildi sem er löglegt viðfang í innfluttu
;;;           samanburðaraðgerðina <<==.
;;;   Eftir:  Búið er að bæta x í p.
;;;
;;;   Notkun: x = remove(p);
;;;   Fyrir:  p er forgangsbiðröð, ekki tóm.
;;;   Eftir:  x er það gildi sem var fremst í p, m.v. innfluttu
;;;           samanburðaraðgerðina <<==.
;;;           Það hefur verið fjarlægt úr p.
;;;
;;;   Notkun: b = isEmpty(p);
;;;   Fyrir:  p er forgangsbiðröð.
;;;   Eftir:  b er satt ef p er tóm, ósatt annars.
;;;
;;; Innflutt
;;; ========
;;;   Notkun: b = x <<== y;
;;;   Fyrir:  x og y eru gildi af þeirri gerð sem við viljum
;;;           geta geymt í forgangsbiðröð.
;;;   Eftir:  b er satt ef x verður að vera á undan y, ósatt
;;;           ef x má ekki vera á undan y.

"pq.mmod" =
{{
;;; Fastayrðing gagna:
;;;    Forgangsbiðröð sem inniheldur gildi x1, x2, ..., xN,
;;;    í engri sérstakri röð, er geymd sem listi [[x1,x2,...,xN]].

makePQ =
  fun()
  {
    [[]];
  };

add =
  fun(p,x)
  {
    setHead(p,x:head(p));
  };
```

```
remove =
  fun(p)
  {
    ...
  };

isEmpty =
  fun(p)
  {
    ...
  };
}}
;
```

10. Skrifið eftirfarandi atriði fyrir biðraðaeiningu í Morpho:

    - hönnunarskjal
    - fastayrðingu gagna
    - aðgerð sem fjarlægir gildi

    Athugið að aðeins þarf að forrita eina aðgerð. Þið hafið frjálsar hendur um fastayrðingu gagna.

    Munið upplýsingahuld.

    Write the following for a queue module in Morpho:

    - a design document,
    - a data invariant,
    - an operation to remove a value

    Note that you only need to implement one operation. You are free to write the data invariant any way you wish (as long as it works).

    Lausn:

```
;;; Útflutt
;;; =======
;;;   Notkun: q = makeQueue();
;;;   Eftir:  q er ný tóm biðröð
;;;
;;;   Notkun: add(q,x);
;;;   Fyrir:  q er biðröð,
;;;           x er gildi.
;;;   Eftir:  Búið er að bæta x aftast í q.
;;;
;;;   Notkun: x = remove(q);
;;;   Fyrir:  q er biðröð, ekki tóm.
;;;   Eftir:  x er það gildi sem var fremst í q,
;;;           það hefur verið fjarlægt úr q.
;;;
;;;   Notkun: b = isEmpty(q);
;;;   Fyrir:  q er biðröð.
;;;   Eftir:  b er satt ef q er tóm, ósatt annars.

"queue.mmod" =
{{
;;; Fastayrðing gagna:
```

```
;;;    Biðröð sem inniheldur gildi x1, x2, ..., xN
;;;    þar sem x1 er á undan x2, o.s.frv., er geymd
;;;    sem listi [[x1,x2,...,xN]].

makeQueue =
  fun()
  {
    [[]];
  };

add =
  fun(p,x)
  {
    ...
  };

remove =
  fun(p)
  {
    val res = head(head(p));
    setHead(p,tail(head(p)));
    res;
  };

isEmpty =
  fun(p)
  {
    ...
  };
}}
;
```