TÖL304G Forritunarmál
Miðannarpróf
TÖL304G Programming Languages
Midterm Exam

Leysið 6 af eftirfarandi dæmum og a.m.k. eitt úr hverjum kafla. Ef þið leysið fleiri en 6 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 6 of the following exercises and at least one from each chapter. If you solve more than 6 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 (MPG, BÞ eða GÖS) á úrlausnarblöðin.

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

Kafli I - Almennt efni

Chapter I - General problems

1. Sýnið BNF, EBNF eða málrit fyrir mál reiknisegða með svigum, breytunafninu a, tvíundaraðgerðunum $+$ og $-$ og einundaraðgerðinni $-$. Dæmi um löglegar formúlur eru þá t.d. a, $(a) + -a$ og $a - a$, en ekki t.d. $a - +a$.

   Show BNF, EBNF or syntax diagrams for a language of expressions with parentheses, the variable name a, the binary operations $+$ and $-$, and the unary operation $-$. Examples of valid expressions include a, $(a) + -a$, and $a - a$, but not, for example, $a - +a$.

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 defined 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.

   (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.

   (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.

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

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

```
  y = 2;
}

int i,a[100];
for( i=0 ; i!=100 ; i++ ) a[i]=i;
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?
- tilvísunarviðföng — call by reference?
- nafnviðföng — call by name?

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 (í hvaða stofnum) 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 ...)
      ...stofn H/body of H...
    )
    ...stofn G/body of G...
  )
  (define (I ...)
    ...stofn I/body of I...
  )
  ...stofn F/body of F...
)
```

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

```
rec fun F(...)
{
    rec fun G(...)
    {
        rec fun H(...)
        {
            ...stofn H/body of H...
        };
        ...stofn G/body of G...
    },
    fun I(...)
    {
        ...stofn I/body of I...
    };
    ...stofn F/body of F...
};
```

   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 (in which bodies) 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.

5. - Hvað innihalda vakningarfærslur? Hver er munurinn á innihaldi vakningarfærslna í bálkmótuðum forritunarmálum, annars vegar, og hins vegar forritunarmálum sem ekki eru bálkmótuð?
   What is the contents of activation records? What is the difference of the contents of activation records in block-structured languages and those that are not block structured.
   - Hvað er lokun og hvaða upplýsingaratriði eru geymdar í lokunum?
   What is a closure and what items of information are stored in closures?

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 margfeldinu af summunum 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 product of the sums of the numbers in the inner lists.

7. Skrifið halaendurkvæmt[1] fall í Scheme, CAML eða Morpho, sem tekur eina heiltölu $n \geq 0$ sem viðfang og skilar listanum $[1, 2, \ldots, n]$.

   Write a tail recursive[2] function in Scheme, CAML or Morpho, that takes as argument one integer $n \geq 0$ and returns the list $[1, 2, \ldots, n]$.

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.

Kafli IV - Einingaforritun

Chapter IV - Modular Programming

9. Skrifið

   - hönnunarskjal,
   - fastayrðingu gagna
   - og smíð (forritun) á aðgerð til að fjarlægja 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 remove 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.

10. Skrifið eftirfarandi atriði fyrir hlaðaeiningu í Morpho:

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

[2] 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.

- 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 stack 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).

Remember information hiding.

TÖL304G Forritunarmál
Miðannarpróf
TÖL304G Programming Languages
Midterm Exam

Leysið 6 af eftirfarandi dæmum og a.m.k. eitt úr hverjum kafla. Ef þið leysið fleiri en 6 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 6 of the following exercises and at least one from each chapter. If you solve more than 6 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 (MPG, BÞ eða GÖS) á úrlausnarblöðin.
No help materials are allowed. The exam counts as one home assignment. Write your name and the initials of your tutor (MPG, BÞ or GÖS) on the solution papers.

Kafli I - Almennt efni

Chapter I - General problems

1. Sýnið BNF, EBNF eða málrit fyrir mál reiknisegða með svigum, breytunafninu a, tvíundaraðgerðunum $+$ og $-$ og einundaraðgerðinni $-$. Dæmi um löglegar formúlur eru þá t.d. a, $(a) + -a$ og $a - a$, en ekki t.d. $a - +a$.

   Show BNF, EBNF or syntax diagrams for a language of expressions with parentheses, the variable name a, the binary operations $+$ and $-$, and the unary operation $-$. Examples of valid expressions include a, $(a) + -a$, and $a - a$, but not, for example, $a - +a$. BNF:
   Lausn:

   <E> ::= a | <E> + <E> | - <E> | (<E>)

   EBNF:

   e = ’a’ | e, ’+’, e | ’-’,e | ’(’, e, ’)’ ;

   Málrit:

   <E>:

   

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 defined 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 = 1;
  print x,y;
  y = 2;
}

int i,a[100];
for( i=0 ; i!=100 ; i++ ) a[i]=i;
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:
  1 0
  0 1
- tilvísunarviðföng — call by reference?
  Lausn:
  1 1
  2 1
- nafnviðföng — call by name?
  Lausn:
  1 1
  1 2

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 (í hvaða stofnum) 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 ...)
      ...stofn H/body of H...
    )
    ...stofn G/body of G...
  )
  (define (I ...)
    ...stofn I/body of I...
  )
  ...stofn F/body of F...
)
```

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

```
rec fun F(...)
{
    rec fun G(...)
    {
        rec fun H(...)
        {
            ...stofn H/body of H...
        };
        ...stofn G/body of G...
    },
    fun I(...)
    {
        ...stofn I/body of I...
    };
    ...stofn F/body of F...
};
```

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 (in which bodies) 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. - Hvað innihalda vakningarfærslur? Hver er munurinn á innihaldi vakningarfærslna í bálkmótuðum forritunarmálum, annars vegar, og hins vegar forritunarmálum sem ekki eru bálkmótuð?
   What is the contents of activation records? What is the difference of the contents of activation records in block-structured languages and those that are not block structured.
   Lausn: Vakningarfærslur innihalda:
   - Viðföng (arguments)
   - Staðværar breytur (local variables)
   - Vendivistfang (return address)
   - Stýrihlekk (control link, dynamic link)
   - Aðgangshlekk (tengihlekk, static link access link)

   Aðgangshlekkurinn er aðeins notaður í bálkmótuðum forritunarmálum (block-structured). Hin atriðin eru í öllum.

- Hvað er lokun og hvaða upplýsingaratriði eru geymdar í lokunum?
  What is a closure and what items of information are stored in closures?
  Lausn: Lokun er gildi í bálkmótuðu forritunarmáli sem stendur fyrir fall sem hægt er að kalla á. Lokun inniheldur fallsbendi og aðgangshlekk.

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 margfeldinu af summunum 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 product of the sums of the numbers in the inner lists.
   Lausn:
   Lausn: (Í CAML Light)

```
(*
Notkun: prodsum x
Fyrir:  x = [[x11;x12;...];...;[xK1;xK2;...]]
        er float list list (listi af listum
        af float).
Gildi:  (x11+x12+...)*...*(xK1+xK2+...)
*)
let prodsum x =
       it_list (prefix *.)
               1.0
               (map (function x->it_list (prefix +.) 0.0 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 (prodsum x)
  (insert * 1 (map (lambda (x) (insert + 0 x)) x))
)
```

   Eða, í Racket:

```
(define (sumprod x)
  (foldl * 0 (map (lambda (x) (foldl + 1 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 = prodsum(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;},0.0,x)},x);
  foldl(fun(x,y){x*y},1.0,x);
};
```

7. Skrifið halaendurkvæmt[1] fall í Scheme, CAML eða Morpho, sem tekur eina heiltölu $n \geq 0$ sem viðfang og skilar listanum $[1, 2, \ldots, n]$.

   Write a tail recursive[2] function in Scheme, CAML or Morpho, that takes as argument one integer $n \geq 0$ and returns the list $[1, 2, \ldots, n]$.
   Lausn (í Scheme):

```
;; Notkun: (index n)
;; Fyrir:  n er heiltala, n>=0
;; Gildi:  Listinn (1 2 ... n)
(define (index n)
  ;; Notkun: (help n x)
  ;; Fyrir:  x=(x1 x2 ...) er listi
  ;;         n er heiltala, n>=0
  ;; Gildi:  Listinn (1 2 ... n x1 x2 ...)
  (define (help n x)
    (if (= n 0)
        x
        (help (- n 1) (cons n x))
    )
  )
  (help n '())
)
```

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.

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

[2] 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.

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:

```
;; Notkun: (map-stream f s)
;; Fyrir:  s=[s1 s2 ...] er óendanlegur straumur.
;;         f er einundarfall.
;; Gildi:  Straumurinn [(f s1) (f s2) ...].
(define (map-stream f s)
  (cons-stream (f (stream-car s))
               (map-stream f (stream-cdr s))
  )
)

;; Notkun: odds
;; Gildi:  Óendanlegi straumurinn af oddatölum [1 3 5 ...]
(define odds
  (cons-stream
    1
    (stream-map
      (lambda (n) (+ 2 n))
      odds
    )
  )
)
```

Kafli IV - Einingaforritun

Chapter IV - Modular Programming

9. Skrifið

   - hönnunarskjal,
   - fastayrðingu gagna
   - og smíð (forritun) á aðgerð til að fjarlægja 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 remove 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 fylki \([x1,x2,...,xN])
;;;    þar sem núllta sætið inniheldur lista af gildunum í
;;;    forgangsbiðröðinni í þeirri röð sem gildin eiga að
;;;    fjarlægjast í, þ.a. x1 er þá fyrsta gildi sem fer út.

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

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

remove =
  fun(p)
  {
    val result = head(p[0]);
    p[0] = tail(p[0]);
  };

isEmpty =
  fun(p)
```

```
  {
    ...
  };
}}
;
```

10. Skrifið eftirfarandi atriði fyrir hlaðaeiningu 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 stack 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).

    Remember information hiding.

```
;;; Útflutt
;;; =======
;;;   Notkun: s = makeStack();
;;;   Eftir:  s er nýr tómur hlaði
;;;
;;;   Notkun: push(s,x);
;;;   Fyrir:  s er hlaði,
;;;           x er gildi.
;;;   Eftir:  Búið er að bæta x ofan á s.
;;;
;;;   Notkun: x = pop(s);
;;;   Fyrir:  s er hlaði, ekki tómur.
;;;   Eftir:  x er það gildi sem var efst á s,
;;;           það hefur verið fjarlægt af s.
;;;
;;;   Notkun: b = isEmpty(s);
;;;   Fyrir:  s er hlaði.
;;;   Eftir:  b er satt ef s er tómur, ósatt annars.

"stack.mmod" =
{{
;;; Fastayrðing gagna:
;;;    Hlaði sem inniheldur gildi x1, x2, ..., xN
;;;    þar sem x1 er efst, x2 næstefst, o.s.frv., er geymdur
;;;    sem bendir á breytu sem inniheldur listann [x1,x2,...,xN].

makeStack =
  fun()
  {
    var svar;
    &svar;
```

```
  };

push =
  fun(s,x)
  {
    ...
  };

pop =
  fun(s)
  {
    val result = head(*s);
    *s = tail(*s);
    res;
  };

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