

# TÖL304G

## Forritunarmál

### Verkefnablað 1

Snorri Agnarsson

25. ágúst 2019

## Skilatími

Einstaklingsverkefnum og hópverkefnum skal skila fyrir kl. 23:59 fimmtudaginn 5. september. Í vikublaði 1 má sjá nánari leiðbeiningar um skil.

Útgöngusvörum dæmatíma skal skila í lok viðkomandi dæmatíma og skila á einni blaðsíðu (sem þið skuluð sjálf leggja til) sem skal vera merkt með nafni og háskóla-tölvupóstfangi, t.d. „Jón Jónsson, nn123@hi.is“.

Það verða engir dæmatímar í fyrstu viku og því er engin útgöngusurning fyrir dæmatíma. Í öðrum vikum verða yfirleitt dæmatímar og þá verður stundum tilgreind útgöngusurning fyrir dæmatímann. Ef engin útgöngusurning er tilgreind þá er sjálfgefin útgöngusurning: „Hver er upphaldsliturinn þinn?“.

## Hópverkefni

Mælt er með því að hópverkefnin séu unnin í dæmatíma.

1. Hvað er mál?
  - (A) Mengi strengja.
  - (B) Strengur.
  - (C) Fall frá strengjum yfir í mengi merkinga.
  - (D) Fall frá heiltölum yfir í strengi.

2. Sýnið BNF, EBNF og málrít fyrir eftirfarandi mál, ef hægt er. Ef eitthvað af þessu er ekki hægt tilgreinið þá hvers vegna það er ekki hægt. Ekki þarf að sanna að það sé ekki hægt, en ástæðan sem þið tilgreinið þarf að vera rétt. Gerið ráð fyrir að  $0 \in \mathbb{N}$ .

a)  $\{a^n b^k c^n \mid n, k \in \mathbb{N}\}$ .

b)  $\{a^k b^n c^n \mid n, k \in \mathbb{N}\}$ .

c)  $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ .

d)  $\{a^n b^n c^k \mid n, k \in \mathbb{N}\}$ .

3. Lýsið í stuttu máli (á íslensku eða ensku) því máli sem eftirfarandi BNF mállýsing skilgreinir. Athugið að þið eigið að lýsa málinu, ekki mállýsingunni.

$$\langle x \rangle ::= a \langle x \rangle \\ \mid \langle y \rangle$$

$$\langle y \rangle ::= b \langle y \rangle \\ \mid \epsilon$$

Sýnið einnig málrít og endanlega stöðuvél fyrir málið. Athugið að endanlega stöðuvélin mun ekki þurfa fleiri en tvær stöður. Endanlega stöðuvélin má vera löggeng eða brigðgeng. Hvort tveggja getur gengið upp.

## Einstaklingsverkefni

1. Hverjar af eftirfarandi fullyrðingum eru réttar?
- (A) Öll samhengisfrjáls mál eru regluleg.
- (B) Öll regluleg mál eru samhengisfrjáls.
- (C) Bæði (A) og (B).
- (D) Hvorki (A) né (B).
2. Íhugið eftirfarandi BNF mállýsingar. Tiltakið hverjar af mállýsingunum lýsa reglulegu máli. Fyrir sérhverja slíka mállýsingu sýnið endanlega stöðuvél og reglulega segð fyrir sama mál.

a)  $\langle x \rangle ::= a \langle x \rangle \\ \mid \langle y \rangle$

$$\langle y \rangle ::= b \langle y \rangle \\ \mid \langle z \rangle$$

$$\langle z \rangle ::= c \langle z \rangle \\ \mid \langle x \rangle \\ \mid \epsilon$$

$$\text{b) } \langle x \rangle ::= \langle x \rangle \text{ a}$$

$$\quad \quad \quad | \quad \langle y \rangle$$

$$\langle y \rangle ::= \langle y \rangle \text{ b}$$

$$\quad \quad \quad | \quad \langle z \rangle$$

$$\langle z \rangle ::= \langle z \rangle \text{ c}$$

$$\quad \quad \quad | \quad \epsilon$$

$$\text{c) } \langle x \rangle ::= \langle x \rangle + \langle x \rangle$$

$$\quad \quad \quad | \quad ( \langle x \rangle )$$

$$\quad \quad \quad | \quad \text{a}$$

$$\text{d) } \langle x \rangle ::= \langle x \rangle + \langle x \rangle$$

$$\quad \quad \quad | \quad \text{a}$$

$$\text{e) } \langle x \rangle ::= ( \langle x \rangle ) \langle x \rangle$$

$$\quad \quad \quad | \quad \epsilon$$

$$\text{f) } \langle x \rangle ::= \langle x \rangle \langle x \rangle +$$

$$\quad \quad \quad | \quad \text{a}$$

3. Sýnið BNF og endanlega stöðuvél og reglulega segð fyrir mál þeirra strengja yfir stafrófið  $\{a,b\}$  þar sem fjöldi  $b$  er slétt tala. Sýnið útleiðslutré fyrir strenginn  $abba$ . Ef fleiri en eitt útleiðslutré kemur til greina samkvæmt mállýsingunni ykkar sýnið þá tvö þeirra.
4. Sýnið BNF, málrit og EBNF fyrir mál segða (*expression*) með breytunafninu  $x$ , tvíundaraðgerðinni  $+$  og svigum. Dæmi um slíkar segðir eru  $x$ ,  $(x)$ ,  $((x))$ ,  $x+x$  og  $x+(x+x)$ , en ekki, til dæmis,  $+x$  og  $((x))$  og  $1+x$ . Tómi strengurinn er að sjálfsögðu ekki í málinu.

# TÖL304G

## Forritunarmál

### Verkefnablað 2

Snorri Agnarsson

1. september 2019

## Efnisyfirlit

1	Skilatími	1
2	Útgönguspurningar	2
3	Hópverkefni	2
4	Einstaklingsverkefni	3

## 1 Skilatími

Hópverkefnum og einstaklingsverkefnum þessum skal skila í Gradescope á tíma sem verður skilgreindur þar.

Þið eigið að prófa öll Scheme verkefnin ykkar í einhverju Scheme kerfi, til dæmis DrRacket og sýna útkomur prófana.

Hér eru nokkur Scheme föll og lykilorð sem mögulegt er að þið viljið nota til að leysa þessi verkefni: `define`, `lambda`, `if`, `and`, `or`, `car`, `cdr`, `cons`, `null?`, `list`, `=`, `*`, `+`.

Athugið að Scheme report<sup>1</sup> inniheldur lýsingar á öllum þessum föllum og lykilorðum og einnig má finna nákvæma skjölun í fylgigögnum fyrir MIT-Scheme og DrRacket.

---

<sup>1</sup><http://www.hi.is/snorri/downloads/r5rs.pdf>

## 2 Útgöngusurningar

**Dæmatími.** Tókst þér að klára bæði einstaklingsverkefnin og hópverkefnin fyrir lok dæmatímans? Svarið „Já“ eða „Nei“.

## 3 Hópverkefni

Í eftirfarandi verkefnum megið þið einungis nota einföldu innbyggðu föllin `car`, `cons`, `cdr`, `null?`, `list` og `*` auk lykilorðanna `lambda`, `define` og `if`. Það ætti ekki að valda vandræðum. Einnig má nota hvaða lesfasta (*literal*) sem verða vill, svo sem `'()` og talnafasta. Þið megið að sjálfsögðu kalla á föllin sem þið skrifið og skilið.

1. Skrifið fall `fcompose` sem tekur tvö viðföng,  $f$  og  $g$  sem hvort tveggja eiga að vera föll sem taka eitt viðfang. Kallið `(fcompose f g)` skal skila samsetta fallinu  $f \circ g$ , þ.e. fallinu  $h$  þar sem  $h(x)$  (í Scheme skrifum við `(h x)`) skilar  $f(g(x))$ . Til dæmis skal `(fcompose sqrt sqrt)` skila falli sem reiknar fjórðu rót. Prófið til dæmis segðina `((fcompose sqrt sqrt) 16)`, sem ætti að skila 2.

Hér er beinagrind:

```
;; Notkun: ((fcompose f g) x)
;; Fyrir: f og g eru einundarföll,
;;        x er löglegt viðfang í g,
;;        (g x) er löglegt viðfang í f.
;; Gildi: (f (g x))
(define (fcompose f g)
  ...
)
```

2. Skrifið Scheme fall `sqall`, sem tekur lista  $(x_1 \dots x_n)$ , sem viðfang, og skilar listanum  $(x_1^2 \dots x_n^2)$ . Til dæmis skal kallið `(sqall (list 1 2 3 4))` skila listanum `(1 4 9 16)`.

Hér er beinagrind:

```
;; Notkun: (sqall x)
;; Fyrir: x=(x1 ... xN) er listi talna.
;; Gildi: Listinn (x1^2 ... xN^2).
(define (squall x)
  ...
)
```

3. Skrifðu halaendurkvæmt<sup>2</sup> Scheme fall `sqallrev`, sem tekur lista  $(x_1 \dots x_n)$ , sem viðfang, og skilar listanum  $(x_n^2 \dots x_1^2)$ . Til dæmis skal kallið `(sqallrev (list 1 2 3 4))` skila listanum `(16 9 4 1)`.

Hér er beinagrind:

```
;; Notkun: (sqallrev x)
;; Fyrir:  x=(x1 ... xN) er listi talna.
;; Gildi:  Listinn (xN^2 ... x1^2).
(define (sqallrev x)
  ;; Notkun: (hjalp ...)
  ;; Fyrir:  ...
  ;; Gildi:  (xN^2 ... x1^2)
  (define (hjalp ...)
    ...
  )
  (hjalp x '())
)
```

4. Skrifðu Scheme fall `myif`, sem tekur tvö viðföng  $x$  og  $y$ , og skilar falli, sem tekur gildi  $z$ , sem viðfang, og skilar  $x$  ef  $z$  er satt og  $y$  ef  $z$  er ekki satt. Til dæmis skal kallið `((myif 1 2) #t)` skila 1, en `((myif 1 2) #f)` skal skila 2.

Hér er beinagrind:

```
;; Notkun: ((myif x y) z)
;; Fyrir:  x og y mega vera hvaða gildi sem er,
;;         z er satt, þ.e. #t, eða ósatt, þ.e. #f.
;; Gildi:  x ef z er satt, annars y.
(define (myif x y)
  ...
)
```

## 4 Einstaklingsverkefni

Klárið Dafny föllin þrjú sem eru ókláruð á Dafny síðunni<sup>3</sup>. Skilið PDF útprenti af lausninni í Gradescope og skilið einnig (fremst í sama útprenti) permalink á lausnina ykkar. Þið getið fengið permalink á lausnina, þegar hún er tilbúin, með því að styðja á hnappinn sem merktur er permalink.

<sup>2</sup>Það dugar að vinnsan gerist í halaendurkvæmu hjálparfalli, þ.e. í fallinu `hjalp` í beinagrindinni hér að neðan. Athugið að halaendurkvæmni samsvarar lykkju í öðrum forritunarmálum og þið ættuð því að íhuga hvernig þið mynduð reiknar útkomuna með lykkju.

<sup>3</sup><https://rise4fun.com/Dafny/wQc7>

# TÖL304G

## Forritunarmál

### Verkefnablað 3

Snorri Agnarsson

8. september 2019

## Hópverkefni

Íhugið eftirfarandi  $\lambda$ -segðir.

- Skrifið Scheme segðir (mega vera Scheme föll) sem jafngilda þeim. Athugið að í  $\lambda$ -reikningi merkir segð  $xy$  fallið  $x$  beitt á viðfang  $y$ , ekki  $x$  margfaldað með  $y$ . Í Scheme myndum við skrifa  $(x\ y)$  til að fá þessa merkingu. Hins vegar leyfum við okkur í  $\lambda$ -reikningi að nota millitákun fyrir reikniðgerðir og þess vegna samsvarar  $\lambda$ -segðin  $x * y$  Scheme segðinni  $(* x\ y)$ . Í  $\lambda$ -reikningi má að ósekju bæta við svigum án þess að merking breytist, en í Scheme þá má hvorki bæta við svigum né fækka þeim án þess að merkingin breytist.
- Ef segðin skilar einföldu gildi (t.d. tölu) skal tiltaka hvert gildið er.
- Ef segðin skilar falli sýnið þá, ef hægt er, hvernig nota má fallið í segð sem skilar einföldu gildi.
- Tiltakið hvaða breytur eru frjálsar í hverri segð (ef einhver er). Athugið að hér er spurt um hvort breytan er frjál í heildarsegðinni, ekki aðeins einhverri undirsegð.
- Endurskrifið einnig  $\lambda$ -segðina og skiptið um breytunöfn þar sem það er hægt án þess að merking hennar breytist og notið breytunöfn  $a$ ,  $b$ , o.s.frv. í stað  $x$ ,  $y$  o.s.frv.

Athugið að hér erum við að nota smá viðbætur við venjulegan  $\lambda$ -reikning, sem eru ansi hefðbundnar. Við leyfum okkur t.d. að skrifa  $x + y$  og ætlumst til að segðin  $5 + 3$  sé umrituð í segðina 8 ef sá möguleiki verður til staðar.

1.  $\lambda x. (\lambda y. (x + y) / y)$
2.  $((\lambda x. (\lambda y. (x + y) / y)) 3) 6$
3.  $((\lambda x. (\lambda y. (x (xy)))) (\lambda x. x^2)) 3$

## Einstaklingsverkefni

Íhugið eftirfarandi  $\lambda$ -segðir.

- Skrifðu Scheme segðir (mega vera Scheme föll) sem jafngilda þeim. Athugið að í  $\lambda$ -reikningi merkir segð  $xy$  fallið  $x$  beitt á viðfang  $y$ , ekki  $x$  margfaldað með  $y$ .
- Ef segðin skilar einföldu gildi (t.d. tölu) skal tiltaka hvert gildið er.
- Ef segðin skilar falli sýnið þá, ef hægt er, hvernig nota má fallið í segð sem skilar einföldu gildi og tiltakið hvaða gildi er útkoman.
- Tiltakið hvaða breytur eru frjálsar í hverri segð (ef einhver er). Athugið að hér er spurt um hvort breytan er frjáls í heildarsegðinni, ekki aðeins einhverri undirsegð.
- Endurskrifið einnig  $\lambda$ -segðina og skiptið um breytunöfn þar sem það er hægt án þess að merking hennar breytist og notið breytunöfn  $a$ ,  $b$ , o.s.frv. í stað  $x$ ,  $y$  o.s.frv.

Athugið að hér erum við að nota smá viðbætur við venjulegan  $\lambda$ -reikning, sem eru nokkuð hefðbundnar þó. Við leyfum okkur t.d. að skrifa  $x + y$  og ætlumst til að segðin  $5 + 3$  sé umrituð í segðina 8 ef sá möguleiki kemur upp.

1.  $\lambda x. ((x + z) / z)$
2.  $\lambda x. (\lambda y. (\lambda z. x (y (yz))))$



# TÖL304G

## Forritunarmál

### Verkefnablað 4

Snorri Agnarsson

15. september 2019

## Verkefni

**Þið skuluð prófa öll verkefnin ykkar í einhverju Scheme kerfi, til dæmis DrRacket og sýna niðurstöður.**

Hér eru nokkur Scheme föll og lykilorð sem mögulegt er að þið viljið nota til að leysa þessi verkefni: `define`, `lambda`, `if`, `and`, `or`, `car`, `cdr`, `cons`, `null?`, `list`, `=`, `*`, `+`.

Athugið að Scheme report<sup>1</sup> inniheldur lýsingar á öllum þessum föllum og lykilorðum og einnig má finna nákvæma skjölun í fylgigögnum fyrir DrRacket og önnur Scheme kerfi.

Athugið að halaendurkvæmni má vera þannig að viðkomandi fall noti hjálparfall til að leysa verkefnið og að hjálparfallið sé halaendurkvæmt.

## Hópverkefni

1. Skriðið halaendurkvæmt Scheme fall `revindex` sem uppfyllir eftirfarandi lýsingu:

```
;; Notkun: (revindex n)
;; Fyrir:  n er heiltala, n>=0
;; Gildi:  listi allra heiltalna i, þannig að
;;         0<i<=n, í minnkandi röð.
```

Til dæmis skal Scheme segðið `(revindex 0)` skila `()` og Scheme segðin `(revindex 5)` skal skila `(5 4 3 2 1)`. Sýnið prófanir, eins og endranær.

---

<sup>1</sup><http://www.hi.is/~snorri/downloads/r5rs.pdf>

## 2. Skrifðu *halaendurkvæmt* Scheme fall *foldr* sem uppfyllir eftirfarandi lýsingu:

```
;; Notkun: (foldr f x u)
;; Fyrir:  f er tvíundarfall, þ.e. fall
;;         sem tekur tvö viðföng af einhverju
;;         tagi, x er listi (x1 ... xN)
;;         gilda af því tagi, u er gildi
;;         af því tagi.
;; Gildi:  (f x1 (f x2 (f ... (f xN u) ...)))
;; Aths.: Með öðrum orðum, ef við skilgreinum
;;         tvíundaraðgerð ! með  $a!b = (f\ a\ b)$ ,
;;         þá er útkoman úr fallinu gildið á
;;          $x1\ !\ x2\ |\ \dots\ !\ xN\ !\ u$ 
;;         þar sem reiknað er frá hægri til
;;         vinstri
```

Til dæmis skal segðin `(foldr + (list 1 2) 3)` skila tölunni 6, þ.e.  $1 + 2 + 3$ , reiknað frá hægri. Athugið að það er nauðsynlegt að tryggja að röð reiknaðgerða sé nákvæmlega rétt þannig að, til dæmis, segðin `(foldr - (list 1 2) 3)` skili réttu gildi samkvæmt lýsingunni að ofan, þ.e. útkomunni úr segðinni `(- 1 (- 2 3))`, sem er 2. Þið munið þurfa hjálparfall, sem þið skuluð skrifa skilmerkilega lýsinu á (þ.e. Notkun/Fyrir/-Gildi). Þið megið nota innbyggða fallið `reverse`, sem snýr við lista. Til hliðsjónar er gagnlegt að kíkja á Dafny fallið `FoldR_loop` sem er hér<sup>2</sup> á vefnum.

Sýnið prófun á segðunum `(foldr - '(2 3) 1)` og `(foldr cons '(1 2) '())` og `(foldr cons '() 1)`.

Athugið að rökstuðningur hjálparfallsins er lykilatriði í þessu dæmi. Ekki fást mörg stig fyrir dæmið ef rökstuðningurinn er gallaður.

## 3. Notið föllin að ofan (`revindex` og `foldr`) til að skrifa tvær Scheme segðir til að reikna summu og margfeldi talnanna $1, \dots, 20$ .

## Einstaklingsverkefni

1. Skrifðu Scheme fall `foldrec` sem hefur sömu lýsingu og `foldr` að ofan, en skal ekki vera *halaendurkvæmt*. Ekki má skrifa nein hjálparföll. Þetta ætti að vera létt verk fyrir þá sem skilja grundvallaratriði í Scheme.

Sýnið prófun á segðunum `(foldrec - '(2 3) 1)` og `(foldrec cons '(1 2) '())` og `(foldrec cons '() 1)`.

---

<sup>2</sup><https://rise4fun.com/Dafny/xR7n>

## 2. Skrifðu halaendurkvæmt Scheme fall `foldl` sem uppfyllir eftirfarandi lýsingu:

```
;; Notkun: (foldl f u x)
;; Fyrir:  f er tvíundarfall, þ.e. fall
;;         sem tekur tvö viðföng af einhverju
;;         tagi, x er listi (x1 ... xN)
;;         gilda af því tagi, u er gildi
;;         af því tagi.
;; Gildi:  (f ... (f (f u x1) x2) ... xN)
;; Aths.: Með öðrum orðum, ef við skilgreinum
;;         tvíundaraðgerð ! með  $a!b = (f a b)$ ,
;;         þá er útkoman úr fallinu gildið á
;;          $u ! x1 ! x2 | \dots ! xN$ 
;;         þar sem reiknað er frá vinstri til
;;         hægri
```

Til hliðsjónar er gagnlegt að kíkja á Dafny fallið `FoldL_loop` sem er hér<sup>3</sup> á vefnum. Munið að halaendurkvæmt fall er hliðstætt lykkju.

Ekki má skrifa nein hjálparföll, enda eru þau algerlega ónauðsynleg í þessu verkefni.

Sýnið prófun á segðunum `(foldl - 1 '(2 3))` og `(foldl (lambda (x y) (cons y x)) '() '(1 2 3))`.

---

<sup>3</sup><https://rise4fun.com/Dafny/xR7n>

# TÖL304G

## Forritunarmál

### Verkefnablað 5

Snorri Agnarsson

23. september 2019

## Verkefni

Munið að föll sem skilað er, þar með talið hjálparföll, hafa skýra og rétta lýsingu með „Notkun: ...“, „Fyrir: ...“ og „Gildi: ...“. Takið eftir að í sumum tilfellum þurfa forskilyrði að innihalda lýsingar á sama sniði fyrir viðföng sem eru föll, og svipað gildir í eftirskilyrði (þ.e. „Gildi: ...“) fyrir gildi sem eru föll.

Mikilvægt er að:

- Allar lausnir séu vel sniðsettar og *með réttri innfellingu*.
- Öll **föll** hafi skýra lýsingu með „Notkun:“, „Fyrir:“ og annaðhvort „Eftir:“ eða „Gildi:“, þ.e. forskilyrði og eftirskilyrði. Sé lýsingin gefin fyrirfram skal nota hana, en athugið að hjálparföll sem þið finnið sjálf upp þurfa sína lýsingu. Athugið samt að þótt gefin sé lýsing falls má skrifa aðra lýsingu ef forskilyrði er víkkað og eftirskilyrði er þrengt. Það kemur fyrir sð slíkt sé gagnlegt, til dæmis ef endurkvæm notkun fallsins krefst betra falls en upphaflega lýsingin krefst.
- Sérhver nemandi skal vinna sín einstaklingsverkefni einn og óháður öðrum nemendum. Samræður milli nemenda um verkefnin eru af hinu góða, en afritun einstaklingsverkefna er að sjálfsögðu óheimil.

Í öllum Scheme verkefnunum er ætlast til að forritunin sé án hliðarverkana nema annað sé tekið fram. Þetta þýðir einfaldlega að ekki er leyfilegt að gefa breytu nýtt gildi eða gera uppskurð á lista. Ef þið notið aldrei neina aðgerð í Scheme sem hefur táknið '!' í sínu nafni þá munuð þið uppfylla þetta skilyrði. Dæmi um slíka forboðna aðgerð er `set !`, sem notuð er til að gefa breytu nýtt gildi.

# Hópverkefni

## 1. Skrifðu fall `mulstreams` með eftirfarandi lýsingu:

```
;; Notkun: (mulstreams x y)
;; Fyrir:  x er óendanlegur straumur talna,
;;         x=[x1 x2 x3 ...].
;;         y er einnig óendanlegur straumur talna,
;;         y=[y1 y2 y3 ...].
;; Gildi:  Óendanlegur straumur óendanlegra strauma
;;         talna sem er
;;         [[x1*y1 x2*y1 x3*y1 ...]
;;          [x1*y2 x2*y2 x3*y2 ...]
;;          [x1*y3 x2*y3 x3*y3 ...]
;;          .
;;          .
;;          .
;;          ]
```

Prófið fallið `mulstreams` með segðinni (`mulstreams heil heil`) og sýnið hluta af útkomunni. Útkoman er óendanleg margföldunartafla þannig að vonlaust er að sjá hana alla, en athugið að ef breytan `s` inniheldur óendanlegan straum af óendanlegum straumum þá mun segðin

```
(map (lambda (x) (stream-list x 5)) (stream-list s 5))
```

skila lista af listum sem eru hornið á fylkinu. Slíkt gildi getur Racket birt okkur á læsilegan hátt.

Vísbending: Stofninn á fallinu `mulstreams` verður eðlilega ein beiting á `cons-stream`. Spyrjið ykkur hver hausinn er á útkomunni og hvernig hægt er að reikna hann (það má gera með einu kalli á fall sem finna má í `straumar.rkt`) og hver halinn er og hvernig hægt er að reikna hann.

## 2. Skrifðu fall `squarestream` með eftirfarandi lýsingu:

```
;; Notkun: (squarestream s)
;; Fyrir:  s er óendanlegur straumur talna,
;;         s=[x1 x2 x3 ...].
;; Gildi:  Óendanlegur straumur óendanlegra strauma
;;         talna sem er
;;         [[x1*x1 x2*x1 x3*x1 ...]
;;          [x1*x2 x2*x2 x3*x2 ...]
;;          [x1*x3 x2*x3 x3*x3 ...]
;;          .
```

```
;; .
;; .
;; ]
```

Prófið fallið `squarestream` á straumnum `heil` og sýnið hluta af útkomunni.

Athugið að þægileg aðferð til að þróa föll sem byggð eru á straumaföllum er að sækja forritstextann fyrir strauma úr Uglunni og bæta ykkar falli aftast í skrána.

### 3. Skrifðu fall `powerlist` sem hefur eftirfarandi lýsingu:

```
;; Notkun: (powerlist x)
;; Fyrir: x er endanlegur listi,
;;        x=(x1 x2 ... xN)
;; Gildi: Listinn (y1 y2 y3 ...)
;;        sem inniheldur alla lista sem
;;        hægt er að smíða með því að taka
;;        núll eða fleiri gildi úr x, í
;;        sömu röð og í x, og skeyta þeim
;;        saman í lista.
```

Athugið að þetta er svipað og að reikna veldismengi mengis. Röð listanna í útkomulistanum er valfrjál. Til dæmis myndi `(powerlist '())` skila listanum `()` (sem samsvarar því að veldismengið af tóamenginu er mengið sem inniheldur aðeins tóamengið. Segðin `(powerlist '(1))` gæti skilað listanum `() (1)` eða listanum `((1) ())`. Segðin `(powerlist '(1 2))` gæti skilað listanum `() (1) (2) (1 2)`.

Fjöldi staka (innri lista) í útkomulistanum ætti ávallt að vera  $2^N$  þar sem  $N$  er fjöldi gilda í viðfanginu  $x$ .

Innbyggðu föllin `map` og `append` eru gagnleg hér. Þetta verkefni er hægt að leysa á tiltölulega einfaldan hátt í Scheme.

## Einstaklingsverkefni

1. Skrifðu endurkvæmt Scheme fall sem samsvarar Dafny fallinu `RealPow_recursive` hér<sup>1</sup>. Skylda er að sjá til þess að dýpt endurkvæmni sé í mesta lagi í hlutfalli við  $\log_2(x)$ , alls ekki í hlutfalli við  $x$ .

Þið munuð væntanlega vilja nota Scheme föllin `remainder` og `quotient`. Fyrir heiltölur  $x$  og  $y$  þannig að  $x, y > 0$  gildir að `(remainder x y)` skilar

<sup>1</sup><https://rise4fun.com/Dafny/SEz8>

afgangnum þegar  $x$  er deilt með  $y$ , og  $(\text{quotient } x \ y)$  skilar útkomunni úr heiltöludeilingu á  $x$  með  $y$ . Prófið þessi föll í Scheme til að sjá hverju þau skila og berið  $\text{quotient}$  saman við  $(/ \ x \ y)$ .

Lýsing fallsins skal vera næganlega skýr til að glöggur lesandi geti sannfært sig um að virknin sé sönnuð án þess að bæta þurfi við forskilyrðum eða eftirskilyrðum titl þess að sönnunin gangi upp. Þið megið reikna með því að glöggur lesandi viti að  $(x^2)^z = x^{2z}$ .

Prófið fallið með því að hefja töluna  $1 + 10^{-10}$  í veldið  $10^{10}$  og sýnið hvernig þið prófið og útkomuna úr prófinu. Þið skuluð sjá til þess að talan sem hafin er í veldi sé fleytitala frekar en ræð tala.

2. Skrifið halaendurkvæmt Scheme fall sem samsvarar Dafny fallinu `RealPow_loop` hér<sup>2</sup>. Skylda er að sjá til þess að dýpt endurkvæmni sé í mesta lagi í hlutfalli við  $\log_2(x)$ , alls ekki í hlutfalli við  $x$ .

Þið munið þurfa halaendurkvæmt hjálparfall til að líkja eftir lykkunni.

Lýsing fallanna skal vera næganlega skýr til að glöggur lesandi geti sannfært sig um að virknin sé sönnuð.

Prófið fallið með því að hefja töluna  $1 + 10^{-10}$  í veldið  $10^{10}$  og sýnið hvernig þið prófið og útkomuna úr prófinu. Þið skuluð sjá til þess að talan sem hafin er í veldi sé fleytitala frekar en ræð tala. Íhugið hvers vegna.

3. Skrifið fall `byltalista` með eftirfarandi lýsingu:

```
;; Notkun: (byltalista z)
;; Fyrir:  z er listi jafnlangra lista,
;;          z = ((x11 x12 ... x1N)
;;              (x21 x22 ... x2N)
;;              (x31 x32 ... x3N)
;;              .
;;              .
;;              .
;;              (xM1 xM2 ... xMN)
;;              )
;; Gildi:  Listinn sem er byltingin
;;         (transpose) af z, þ.e.
;;         ((x11 x21 ... xM1)
;;         (x12 x22 ... xM2)
;;         (x13 x23 ... xM3)
;;         .
```

<sup>2</sup><https://rise4fun.com/Dafny/SEz8>

```

; ;      .
; ;      .
; ;      (x1N x2N ... xMN)
; ;      )

```

Athugið að fallið þarf ekki að virka ef innri listarnir eru ekki jafnlangir því samkvæmt forskilyrði á slíkt ekki að gerast. Ef innri listarnir eru tómir er eðlilegt að fallið skili tóma listanum. Sama gildir ef ytri listinn er tómur. Munið að sýna prófanir.



# TÖL304G

## Forritunarmál

### Verkefnablað 6

Snorri Agnarsson

7. október 2019

## Verkefni

### Lýsingar falla

Í öllum verkefnunum skulu öll föll, þar með talið hjálparföll, hafa skýra og rétta lýsingu með „Notkun: ...“, „Fyrir: ...“ og „Gildi: ...“. Takið eftir að í sumum tilfellum þurfa forskilyrði að innihalda lýsingar á sama sniði fyrir viðföng sem eru föll, og svipað gildir í eftirskilyrði (þ.e. „Gildi: ...“) fyrir gildi sem eru föll.

### Prófanir falla

Munið að alltaf skal sýna einhverjar ófáfengilegar prófanir á öllum föllum sem þið skilið. Ekki þarf að sýna skjáskot af prófunum, en sýna skal hvernig prófanirnar eru forritaðar og hverjar útkomurnar eru.

Gott getur verið að ræða um prófanir ykkar á milli og á Piazza, einnig fyrir einstaklingsverkefni.

### Hópverkefni

Fyrir þessi verkefni skuluð þið sækja skrána `permutations.ml-beinagrind` úr Uglunni og vista hana hjá ykkur undir nafninu `permutations.ml`. Gerið síðan viðeigandi viðbætur og sýnið forritstextannn ásamt lýsingum fyrir föllin sem þið klárið.

1. Skrifðu fall `mapreduce` í CAML, þar sem kallið `mapreduce f op u x` tekur fall `f`, tvíundaraðgerð `op`, gildi `u` og lista `x = [x1; ...; xn]` og skilar gildinu  $u \oplus f(x_1) \oplus \dots \oplus f(x_n)$ , þar sem  $x \oplus y = (op\ x\ y)$ . Fallið skal vera halaendurkvæmt og skal reikna frá vinstri til hægri.

Sýnið eftirfarandi prófun á fallinu:

```
let
  inc x = x+1
and
  add x y = x+y
in
  mapreduce inc add 0 [0;1;2;3;4];;
```

2. Klárið að forrita fallið `fromTo`. Munið að sýna prófanir.
3. Klárið að forrita fallið `insertAt`. Munið að sýna prófanir.
4. Klárið að forrita fallið `extendPermutation`. Munið að sýna prófanir.
5. Klárið að forrita fallið `length`. Munið að sýna prófanir.
6. Þegar þessu er öllu lokið, sýnið útkomuna úr segðinni

```
length (permutations 6);;
```

sem er aftast í skránni ykkar.

## Einstaklingsverkefni

Skrifuðu fall `powerList` í CAML sem tekur heiltölu  $n \geq 0$  sem viðfang og skilar lista af listum sem inniheldur alla mögulega lista sem eru undirlistar listans  $[n; n-1; \dots; 2; 1]$  í þeim skilningi að undirlistarnir innihalda gildi úr listanum  $[n; n-1; \dots; 2; 1]$  í sömu röð og í  $[n; n-1; \dots; 2; 1]$ , nema hvað fjarlægja má núll eða fleiri gildi úr listanum  $[n; n-1; \dots; 2; 1]$ .

Til dæmis myndi `powerList 0` skila  `[[] ]` og `powerList 2` gæti skilað  `[[]; [2]; [1]; [2;1] ]`, eða lista sömu lista í annarri röð.

Þið megið nota eins mörg hjálparföll eins og þið viljið og nota hvaða innbyggð föll í CAML sem ykkur hentar. Athugið að tvíundaraðgerðin `@` skeytir saman tveimur listum.

# TÖL304G

## Forritunarmál

### Verkefnablað 7

Snorri Agnarsson

6. október 2019

## Verkefni

### Lýsingar falla

Í öllum verkefnunum skulu öll föll, þar með talið hjálparföll, hafa skýra og rétta lýsingu með „Notkun: ...“, „Fyrir: ...“ og „Gildi: ...“. Takið eftir að í sumum tilfellum þurfa forskilyrði að innihalda lýsingar á sama sniði fyrir viðföng sem eru föll, og svipað gildir í eftirskilyrði (þ.e. „Gildi: ...“) fyrir gildi sem eru föll.

### Prófanir falla

Munið að alltaf skal sýna einhverjar ófáfengilegar prófanir á öllum föllum sem þið skilið. Ekki þarf að sýna skjáskot af prófunum, en sýna skal hvernig prófanirnar eru forritaðar og hverjar útkomurnar eru.

Gott getur verið að ræða um prófanir ykkar á milli og á Piazza, einnig fyrir einstaklingsverkefni.

### Listi úr óendanlegum straumi

Eftirfarandi fall getur verið hjálplegt í prófunum.

```
;;; Notkun: x = take(n,s);  
;;; Fyrir:  n er heiltala >=0,  
;;;       s er óendanlegur straumur.  
;;; Eftir:  x er listi sem inniheldur n-1
```

```

;;;          fremstu gildi úr s í sömu röð
;;;          og í s.
rec fun take(n,s)
{
  if( n==0 ) {return []};
  return streamHead(s) : take(n-1,streamTail(s));
};

```

## Hópverkefni

1. Skrifið fall `filter` í Morpho sem tekur tvö viðföng, fall  $p$  og lista  $x = [x_1, \dots, x_n]$  og skilar lista þeirra gilda  $x_i$  í  $x$  þannig að kallið  $p(x_i)$  skilar „satt“, þ.e. þannig að skilagildið er hvorki `null` né `false`. Til dæmis ætti segðin `filter(fun(x){x%2==0}, [1,2,3,4,5])` að skila listanum `[2,4]`.
2. Skrifið fall `filterStream` í Morpho sem tekur tvö viðföng, fall  $p$  og óendanlegan straum  $x = [x_1, \dots]$  og skilar straumi þeirra gilda  $x_i$  í  $x$  þannig að kallið  $p(x_i)$  skilar „satt“, þ.e. þannig að skilagildið er hvorki `null` né `false`.
3. Skrifið endurkvæma skilgreiningu á fasta í Morpho

```

;;; Notkun: primes
;;; Fyrir: Ekkert
;;; Gildi: Straumur allra prímtalna án
;;;          endurtekninga í vaxandi röð,
;;;          þ.e. #[2,3,5,7,11,13,17,...].
rec val primes=...;

```

þannig að breytan `primes` innihaldi vaxandi óendanlegan straum sem inniheldur allar prímtölurnar og ekkert annað, þ.e. `#[2,3,5,7,11,13,17,...]`. Þið megið nota fallið `filterStream` hér að ofan og önnur hjálparföll að vild (með réttum lýsingum, að sjálfsögðu).

Vísbending 0: Til þess að allt þetta gangi snurðulaust fyrir sig er ráðlegt að sjá til þess að allar heiltölurnar í straunum séu *stórar heiltölur*, þ.e. heiltölur sem búið er að breyta á `BigInteger` snið. Stóra heiltalan sem samsvarar 12345 er `bigInteger(12345)`. Reikniaðgerðir á stórar heiltölur skila stórum heiltölum. Ef þetta er ekki gert þá munu straumarnir byrja á réttum gildum en fyrr eða síðar munu koma röng gildi, því réttu gildin þurfa að vera stórar heiltölur.

Vísbending 1: Straumurinn `#[2,3,5,7,9,11,13,15,17,...]` getur verið gagnlegur til að byggja á. Ef þið síið úr þeim straumi einungis þær tölur

þar sem engin fremri tala gengur upp í töluna þá fáið þið straum allra prímtalna. Sama má einnig gera með strauminn  $\#[2, 3, 4, 5, 6, 7, 8, \dots]$ , sem er e.t.v. aðeins einfaldara.

Vísbending 2: Það getur verið hjálplegt að skrifa fyrst hjálparfall með eftirfarandi lýsingu:

```
;;; Notkun: p = erPrím(s,n);  
;;; Fyrir:  n er heiltala >1, s er óendanlegur  
;;;        vaxandi straumur mismunandi stórra  
;;;        heiltalna >1 sem inniheldur allar  
;;;        prímtölur sem ganga upp í n.  
;;; Eftir:  p er satt þá og því aðeins að n  
;;;        sé prímtala.
```

Vísbending 3: Eðlilegt er að útfæra `erPrím` með `lykkju` (ásamt `fastayrðingu` `lykkju`), eða (þá þarf ekki `fastayrðingu` `lykkju`) með `halaendurkvæmni`. Í báðum tilvikum stytum við `s` í hverju skrefi.

Vísbending 4: Til þess að gera útreikningana hraðvirkari og einnig til að sjá til þess að möguleg gagnkvæm endurkvæmni lendi ekki í vandræðum þá er gagnlegt að athuga, inni í útfærslunni á `erPrím`, hvort `streamHead(s)*streamHead(s) > n`. Ef svo er þá er `n` áreiðanlega prímtala (íhugið hvers vegna það er satt og íhugið einnig hvers vegna þetta hraðar útreikningunum og hve mikið).

## Einstaklingsverkefni

1. Skrifðu fall `powerList` sem tekur heiltölu  $n \geq 0$  sem viðfang og skilar lista af listum heiltalna sem inniheldur alla mögulega lista heiltalna sem eru undirlistar listans  $x = [n, n-1, \dots, 2, 1]$  í þeim skilningi að undirlistarnir innihalda gildi úr listanum  $x$  í sömu röð og í  $x$ , nema hvað fjarlægja má núll eða fleiri gildi úr listanum  $x$ .

Til dæmis myndi `powerList(0)` skila  `[[] ]` og `powerList(2)` gæti skilað  `[ [], [2], [1], [2, 1] ]`, eða lista sömu lista í annarri röð.

Þið megið nota eins mörg hjálparföll eins og þið viljið og nota hvaða innbyggð föll í `Morpho` sem ykkur hentar.

2. Skrifðu fall `zipWith` í `Morpho` sem tekur fall  $f$  (lokun) og tvo jafnlanga lista,  $x = [x_1, \dots, x_n]$  og  $y = [y_1, \dots, y_n]$  sem viðföng og skilar listanum  $[f(x_1, y_1), \dots, f(x_n, y_n)]$ . Til dæmis ætti kallið `zipWith(fun(x,y){x+y}, [1,2,3], [4,5,6])` að skila `[5,7,9]`.

# TÖL304G

## Forritunarmál

### Verkefnablað 8

Snorri Agnarsson

13. október 2019

#### Skilatími

Í öllum verkefnunum skulu öll föll, þar með talið hjálparföll, hafa skýra og rétta lýsingu með „Notkun: ...“, „Fyrir: ...“ og „Gildi: ...“. Takið eftir að í sumum tilfellum þurfa forskilyrði að innihalda lýsingar á sama sniði fyrir viðföng sem eru föll, og svipað gildir í eftirskilyrði (þ.e. „Gildi: ...“) fyrir gildi sem eru föll.

Athugið að í fjölnota einingum, svo sem forgangsbiðraðaeiningum eins og hér að neðan, þurfa innflutt föll og aðgerðir einnig að hafa slíka lýsingu.

#### Verkefni

##### Hópverkefni

Skilið eftirfarandi:

1. Hönnunarskjali fyrir fjölnota forgangsbiðröð í Morpho. Útfluttar aðgerðir (stef) skulu meðal annars bjóða upp á eftirfarandi:
  - Að smíða nýja tóma forgangsbiðröð. Þetta gæti verið fall sem tekur ekkert viðfang og skilar forgangsbiðröð af ótakmarkaðri stærð. Þetta gæti líka verið fall sem tekur heiltöluviðfang sem væri hámarksstærð forgangsbiðraðarinnar.
  - Að athuga hvort gefin forgangsbiðröð er tóm.
  - Að athuga hvort gefin forgangsbiðröð er full. Þessi aðgerð er ekki nauðsynleg ef allar forgangsbiðraðir eru af ótakmarkaðri stærð.

- Að bæta tilteknu gildi inn í tiltekna forgangsbiðröð með tilteknum forgangi (lykli). Þetta skal vera aðgerð sem tekur tvö viðföng, annað viðfangið er gildið, hitt er lykillinn.
- Að fjarlægja og skila *fremsta* gildi úr tiltekinni forgangsbiðröð (þetta gætu verið tvær eða fleiri mismunandi aðgerðir). Fremsta gildið er það gildi sem hefur lykil með mesta forgang.

Forgangsbiðröðin skal vera útfærð sem ein eining í Morpho og skal samiburðaraðgerð fyrir lykla (forganga) vera innflutt (ótengd). Einingin skal vera *fjölnota* þannig að tengja megi mismunandi samiburðaraðgerðir til að forgangsraða gildum með mismunandi gerðum lykla svo sem heiltölum, fleytitölum og strengjum, svo eitthvað sé nefnt.

2. Útfærslu (smíð, forritstexti) fyrir forgangsbiðröðina sem lýst er í hönnunarskjalinu. Smíðin skal hafa innri skjölun í formi fastayrðingar gagna, sem lýsir því hvernig forgangsbiðraðir eru geymdar í minni tölvunnar.
3. Tvö prófunarforrit (forritstexti) fyrir forgangsbiðraðirnar. Forritin skulu nota forgangsbiðröð til að raða tölum, annars vegar í vaxandi röð og hins vegar í minnkandi röð.

Forritin skulu skrifa runu talna í vaxandi eða minnkandi röð, eftir atvikum, og sýna skal útkomur prófana.

Athugið að það er ekkert því til fyrirstöðu að lykill og samsvarandi gildi séu það sama í forgangsbiðröðinni. Það er þægileg aðferð þegar við notum forgangsbiðröð til að raða.

## Einstaklingsverkefni

Skrifið Morpho einingu fyrir hlaða (*stack*) gilda af hvaða tagi sem er. Skila skal eftirfarandi:

1. Hönnunarskjal sem inniheldur lýsingar á öllum útfluttum og innfluttum (ef einhverjar eru) aðgerðum (föllum) í einingunni. Lýsingarnar felast í liðunum Notkun:..., Fyrir:... og Eftir:....
2. Fastayrðing gagna fyrir hlaðana.
3. Forritstexti fyrir eininguna.
4. Forritstexti fyrir prófunarforrit.
5. Úttak úr prófunarforriti.

# Innsetning gildis í lista

Eftirfarandi getur verið gagnlegt fyrir forgangsbiðröðina. Þennan forritstexta má finna í Uglunni (`insert.morpho`). Athugið að forgangsbiðröð sem notar fall líkt þessu sem sinn kjarna verður varla hraðvirk, en ekki er beðið um hraðvirka forgangsbiðröð. Hraðvirka forgangsbiðröð mætti útfæra með því að geyma pörin (lykill,gildi) í hrúgu (heap), en það krefst þess að annaðhvort sé hver forgangsbiðröð af fastri hámarksstærð eða að fylkinu sem skipað er í hrúgu sé endurúthlutað þegar minnisrými þess þrýtur.

```
1 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2 ;;; Hönnunarskjal fyrir eininguna "insert" ;;;;;;;;;;;;;;;;;
3 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4 ;;;
5 ;;; Innflutt (Imported):
6 ;;; Notkun: z = x <== y
7 ;;; Fyrir: x og y eru gildi af þeirri gerð sem við
8 ;;; viljum bera saman
9 ;;; Eftir: z er satt ef x verður að vera á undan y,
10 ;;; ósatt ef y verður að vera á undan x.
11 ;;; Ath.: Skilagildið má vera hvað sem er ef x má
12 ;;; vera á undan y OG y má vera á undan x.
13 ;;;
14 ;;; Útflutt (Exported):
15 ;;; Notkun: y = insert(x,u)
16 ;;; Fyrir: x er listi gilda af þeirri gerð sem <==
17 ;;; ræður við, í vaxandi röð miðað við <==,
18 ;;; u er einnig gildi af þeirri gerð.
19 ;;; Eftir: y er listi í vaxandi röð miðað við <==
20 ;;; sem inniheldur öll gildin úr x auk u.
21 ;;;
22 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
23 ;;; Hönnunarskjali lýkur hér ;;;;;;;;;;;;;;;;;
24 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
25
26 "insert.mmod" =
27 !
28 {{
29 insert =
30   fun(x,u)
31   {
32     x==[] && (return [u]);
33     head(x) <== u || (return u:x);
34     head(x) : insert(tail(x),u);
35   };
36 }}
37 ;
38
39 show "insert.mmod";
40
41 "test1.mexe" = main in
```



```

42 {{
43 main =
44     fun()
45     {
46         var x=[1,9,2,8,3,7,4,6,5],y;
47         while( x!=[ ] )
48         {
49             y = insert(y,head(x));
50             x = tail(x);
51         };
52         writeln(y);
53     };
54 }}
55 *
56 "insert.mmod"
57 *
58 {{
59 <<== = fun <=(x,y);
60 }}
61 *
62 BASIS
63 ;
64
65 "test2.mexe" = main in
66 {{
67 main =
68     fun()
69     {
70         var x=[1,9,2,8,3,7,4,6,5],y;
71         while( x!=[ ] )
72         {
73             y = insert(y,head(x));
74             x = tail(x);
75         };
76         writeln(y);
77     };
78 }}
79 *
80 "insert.mmod"
81 *
82 {{
83 <<== = fun >(x,y);
84 }}
85 *
86 BASIS
87 ;

```

# Fjarlæging stærsta gildis úr lista

Eftirfarandi getur einnig verið gagnlegt fyrir forgangsbiðröðina. Þennan forritstexta má einnig finna í Uglunni (`removemax.morpho`). Forgangsbiðröð sem notar þetta fall verður varla hraðvirk.

```
1 {;;;
2
3 Hönnunarskjal
4
5 Útfluttar aðgerðir
6 =====
7
8 Notkun: y = removeMax(x);
9 Fyrir: x er listi, ekki tómur, sem inniheldur gildi
10 sem bera má saman með innfluttu aðgerðinni >.
11 Eftir: y er listi sem inniheldur öll gildin úr x nema
12 það gildi sem er stærst miðað við >. Það er
13 þá ekkert annað stak z í listanum x þannig að
14 z>y er satt (nema, reyndar megum við leyfa að
15 z>y ef z og y eru sama gildi).
16
17 Innfluttar aðgerðir
18 =====
19
20 Notkun: b = x>y;
21 Fyrir: x og y eru gildi af því tagi sem við viljum bera
22 saman, þ.e. gildi af því tagi sem mega vera í
23 inntakslistanum x í útfluttu removeMax aðgerðinni.
24 Eftir: b er satt ef x er stærra en y, annars ósatt.
25 Athugasemd: Þessi aðgerð verður að virka sem samanburðaraðgerð,
26 þ.e. verður að vera gegnvirk og svo framvegis. Sérhver
27 tvö gildi þurfa að vera samanburðarhæf þannig að annað
28 gildið sé stærra en hitt, nema gildin séu jöfn. Bæði
29 má nota samanburðaraðgerðir sem virka eins og < og >,
30 eða samanburðaraðgerðir sem virka eins og <= og >=.
31 Samanburðaraðgerðin má því vera andsamhverf (eins og >=)
32 eða ósamhverf (eins og >).
33
34 };;;
35
36 "removemax.mmod" =
37 {{
38 removeMax =
39     fun(x)
40     {
41         var prev=null, currMax=head(x), z=x;
42         while( tail(z) )
43         {
44             ;; z vísar á einhvern hala á listanum x,
```

```

45      ;;; eins og hér er sýnt:
46      ;;;
47      ;;;  x=[x1,x2,...,xI,...,xN]
48      ;;;          z=[xI,...,xN]
49      ;;;
50      ;;; currMax er stærsta gildið af x1,...,xI.
51      ;;; Ef currMax er x1 þá er prev==null, annars
52      ;;; vísar prev á næsta hlekk fyrir framan þann
53      ;;; sem inniheldur currMax.
54      ;;;
55      if( head(tail(z))>currMax )
56      {
57          prev = z;
58          currMax = head(tail(z));
59      };
60      z = tail(z);
61  };
62  if( !prev ) { return tail(x) };
63  setTail(prev,tail(tail(prev)));
64  x;
65  };
66  }}
67  ;
68
69  "test1.mexe" = main in
70  !
71  {{
72
73  ;;; Notkun: y = max(x);
74  ;;; Fyrir:  x=[x1,x2,...,xN] er listi
75  ;;;          gilda sem samanburðaraðgerðin >
76  ;;;          virkar fyrir.
77  ;;; Eftir:  y er stærsta gildið í x miðað
78  ;;;          við samanburðaraðgerðina >.
79  max =
80      fun(x)
81      {
82          var mx = head(x);
83          var z = x;
84          for(;;)
85          {
86              ;;; z vísar á einhvern hala á listanum x,
87              ;;; eins og hér er sýnt:
88              ;;;
89              ;;;  x=[x1,x2,...,xI,...,xN]
90              ;;;          z=[xI,...,xN]
91              ;;;
92              ;;; mx er stærsta gildið af x1,...,xI.
93              ;;;
94              z = tail(z);

```

```

95         if( !z ) { return mx };
96         if( head(z)>mx ) { mx=head(z) };
97     };
98 };
99
100 ;;; Notkun: main();
101 ;;; Fyrir: Ekkert.
102 ;;; Eftir: Búið er að skrifa út tölurnar
103 ;;;      9, 8, 7, 6, 5, 4, 3, 2, 1, eina
104 ;;;      í hverri línu.
105 main =
106     fun()
107     {
108         var x = [1,9,2,8,3,7,4,6,5];
109         while( x )
110         {
111             writeln(max(x));
112             x = removeMax(x);
113         };
114     };
115 }}
116 *
117 "removemax.mmod"
118 *
119 BASIS
120 ;

```

Þessi tvö föll geta, sem sagt, verið gagnleg í einfaldri útfærslu forgangsbiðraða. Hvert um sig er gagnlegt fyrir tiltekna fastayrðingu gagna, sem rætt verður í fyrirlestri. Hér er um tvær mismunandi fastayrðingar gagna að ræða þannig að aðeins annað fallanna verður gagnlegt.