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 háskólatölvupóstfangi, t.d. „Jón Jónsson, nn123@hi.is“.

Það verða engir dæmatímar í fyrstu viku og því er engin útgönguspurning fyrir dæmatíma. Í öðrum vikum verða yfirleitt dæmatímar og þá verður stundum tilgreind útgönguspurning fyrir dæmatímann. Ef engin útgönguspurning er tilgreind þá er sjálfgefin útgönguspurning: „Hver er uppáhaldsliturinn 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 málrit 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) $\{a^n b^k c^n | n, k \in \mathbb{N}\}$.

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

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

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

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

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

   $\langle y \rangle$ ::= b $\langle y \rangle$
   | ϵ

   Sýnið einnig málrit 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 míllýsingar. Tiltakið hverjar af míllýsingunum lýsa reglulegu máli. Fyrir sérhverja slíka míllýsingu sýnið endanlega stöðuvél og reglulega segð fyrir sama mál.

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

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

   $\langle z \rangle$ ::= c $\langle z \rangle$
   | $\langle x \rangle$
   | ϵ

b) $\langle x \rangle$ ::= $\langle x \rangle$ a
| $\langle y \rangle$

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

$\langle z \rangle$ ::= $\langle z \rangle$ c
| ϵ

c) $\langle x \rangle$ ::= $\langle x \rangle$ + $\langle x \rangle$
| ($\langle x \rangle$)
| a

d) $\langle x \rangle$ ::= $\langle x \rangle$ + $\langle x \rangle$
| a

e) $\langle x \rangle$ ::= ($\langle x \rangle$) $\langle x \rangle$
| ϵ

f) $\langle x \rangle$ ::= $\langle x \rangle$ $\langle x \rangle$ +
| 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

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[1] inniheldur lýsingar á öllum þessum föllum og lykilorðum og einnig má finna nákvæma skjölun í fylgigögnum fyrir MIT-Scheme og DrRacket.

[1] http://www.hi.is/ snorri/downloads/r5rs.pdf

2 Útgönguspurningar

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 $(x_1 \ldots x_n)$, sem viðfang, og skilar listanum $(x_1^2 \ldots x_n^2)$. 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. Skrifið halaendurkvæmt[2] Scheme fall `sqallrev`, sem tekur lista (x_1 $\ldots x_n$), sem viðfang, og skilar listanum (x_n^2 $\ldots x_1^2$). 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 (squallrev x)
    ;; Notkun: (hjalp ...)
    ;; Fyrir:  ...
    ;; Gildi:  (xN^2 ... x1^2)
    (define (hjalp ...)
        ...
    )
    (hjalp x '())
)
```

4. Skrifið 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[3]. 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.

[2] Það dugar að vinnslan 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.

[3] https://rise4fun.com/Dafny/wQc7

TÖL304G
Forritunarmál
Verkefnablað 3

Snorri Agnarsson

8. september 2019

Hópverkefni

Íhugið eftirfarandi λ-segðir.

- Skrifið Scheme segðir (mega vera Scheme föll) sem jafngilda þeim. Athugið að í λ-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 λ-reikningi að nota millitáknun fyrir reikniaðgerðir og þess vegna samsvarar λ-segðin $x * y$ Scheme segðinni `(* x y)`. Í λ-reikningi m+a 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 frjáls í heildarsegðinni, ekki aðeins einhverri undirsegð.
- Endurskrifið einnig λ-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 λ-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. $\left((\lambda x.\,(\lambda y.\,(x\,(xy))))\,\left(\lambda x.x^2\right)\right)3$

Einstaklingsverkefni

Íhugið eftirfarandi λ-segðir.

- Skrifið Scheme segðir (mega vera Scheme föll) sem jafngilda þeim. Athugið að í λ-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 λ-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 λ-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[1] 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. Skrifið 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.

[1] http://www.hi.is/~snorri/downloads/r5rs.pdf

2. Skrifið *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 | ... ! 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 reikniaðgerða 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[2] 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 $1, \ldots, 20$.

Einstaklingsverkefni

1. Skrifið 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)`.

[2] https://rise4fun.com/Dafny/xR7n

2. Skrifið 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 | ... ! xN
;;         þar sem reiknað er frá vinstri til
;;         hægri
```

Til hliðsjónar er gagnlegt að kíkja á Dafny fallið `FoldL_loop` sem er hér[3] 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))`.

[3] 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. Skrifið 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. Skrifið 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 straumaföllunum er að sækja forritstextann fyrir strauma úr Uglunni og bæta ykkar falli aftast í skrána.

3. Skrifið 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 valfrjáls. Til dæmis myndi `(powerlist '())` skila listanum `(())` (sem samsvarar því að veldismengið af tómamenginu er mengið sem inniheldur aðeins tómamengið. 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. Skrifið endurkvæmt Scheme fall sem samsvarar Dafny fallinu `RealPow_recursive` hér[1]. 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

[1] https://rise4fun.com/Dafny/SEz8

afgangnum þegar `x` er deilt með `y`, og `(quotient x y)` skilar útkomunni úr heiltöludeilingu á `x` með `y`. Prófið þessi föll í Scheme til að sjá hverju þau skila og berið `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 $\left(x^2\right)^z = x^{2z}$.

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[2]. 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)
;;          .
```

[2] 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

29. september 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. Skrifið fall mapreduce í CAML, þar sem kallið mapreduce f op u x tekur fall f, tvíundaraðgerð op, gildi u og lista x = $[x_1; ...; x_n]$ og skilar gildinu $u \oplus f(x_1) \oplus \ldots \oplus f(x_n)$, 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

Skrifið 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 $[n; n-1; \ldots; 2; 1]$ í þeim skilningi að undirlistarnir innihalda gildi úr listanum $[n; n-1; \ldots; 2; 1]$ í sömu röð og í $[n; n-1; \ldots; 2; 1]$, nema hvað fjarlægja má núll eða fleiri gildi úr listanum $[n; n-1; \ldots; 2; 1]$.

Til dæmis myndi `powerList 0` skila `[[]]` og `powerList 2` gæti skilað `[[];[2];[1];[1;2]]`, 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 $x = [x_1, \ldots, x_n]$ 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 $x = [x_1, \ldots]$ 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 straumunum 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,...]`, 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 styttum 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. Skrifið 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 $x = [n, n-1, \ldots, 2, 1]$ 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. Skrifið fall `zipWith` í Morpho sem tekur fall f (lokun) og tvo jafnlanga lista, $x = [x_1, \ldots, x_n]$ og $y = [y_1, \ldots, y_n]$ sem viðföng og skilar listanum $[f(x_1, y_1), \ldots, f(x_n, y_n)]$. 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 samanburðaraðgerð fyrir lykla (forganga) vera innflutt (ótengd). Einingin skal vera *fjölnota* þannig að tengja megi mismunandi samanburðaraðgerðir 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 einingunа.

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.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Hönnunarskjal fyrir eininguna "insert" ;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;; Innflutt (Imported):
;;;  Notkun: z = x <<== y
;;;  Fyrir:  x og y eru gildi af þeirri gerð sem við
;;;          viljum bera saman
;;;  Eftir:  z er satt ef x verður að vera á undan y,
;;;          ósatt ef y verður að vera á undan x.
;;;  Ath.:   Skilagildið má vera hvað sem er ef x má
;;;          vera á undan y OG y má vera á undan x.
;;;
;;; Útflutt (Exported):
;;;  Notkun: y = insert(x,u)
;;;  Fyrir:  x er listi gilda af þeirri gerð sem <<==
;;;          ræður við, í vaxandi röð miðað við <<==,
;;;          u er einnig gildi af þeirri gerð.
;;;  Eftir:  y er listi í vaxandi röð miðað við <<==
;;;          sem inniheldur öll gildin úr x auk u.
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Hönnunarskjali lýkur hér ;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

"insert.mmod" =
!
{{
insert =
    fun(x,u)
    {
        x==[] && (return [u]);
        head(x) <<== u || (return u:x);
        head(x) : insert(tail(x),u);
    };
}}
;

show "insert.mmod";

"test1.mexe" = main in
```

```
{{
main =
    fun()
    {
        var x=[1,9,2,8,3,7,4,6,5],y;
        while( x!=[] )
        {
            y = insert(y,head(x));
            x = tail(x);
        };
        writeln(y);
    };
}}
*
"insert.mmod"
*
{{
<<== = fun <=(x,y);
}}
*
BASIS
;

"test2.mexe" = main in
{{
main =
    fun()
    {
        var x=[1,9,2,8,3,7,4,6,5],y;
        while( x!=[] )
        {
            y = insert(y,head(x));
            x = tail(x);
        };
        writeln(y);
    };
}}
*
"insert.mmod"
*
{{
<<== = fun >(x,y);
}}
*
BASIS
;
```

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.

```
{;;;

Hönnunarskjal

Útfluttar aðgerðir
==================

Notkun: y = removeMax(x);
Fyrir:  x er listi, ekki tómur, sem inniheldur gildi
        sem bera má saman með innfluttu aðgerðinni >.
Eftir:  y er listi sem inniheldur öll gildin úr x nema
        það gildi sem er stærst miðað við >.  Það er
        þá ekkert annað stak z í listanum x þannig að
        z>y er satt (nema, reyndar megum við leyfa að
        z>y ef z og y eru sama gildi).

Innfluttar aðgerðir
===================

Notkun: b = x>y;
Fyrir:  x og y eru gildi af því tagi sem við viljum bera
        saman, þ.e. gildi af því tagi sem mega vera í
        inntakslistanum x í útfluttu removeMax aðgerðinni.
Eftir:  b er satt ef x er stærra en y, annars ósatt.
Athugasemd: Þessi aðgerð verður að virka sem samanburðaraðgerð,
        þ.e. verður að vera gegnvirk og svo framvegis. Sérhver
        tvö gildi þurfa að vera samanburðarhæf þannig að annað
        gildið sé stærra en hitt, nema gildin séu jöfn.  Bæði
        má nota samanburðaraðgerðir sem virka eins og < og >,
        eða samanburðaraðgerðir sem virka eins og <= og >=.
        Samanburðaraðgerðin má því vera andsamhverf (eins og >=)
        eða ósamhverf (eins og >).

;;;}

"removemax.mmod" =
{{
removeMax =
    fun(x)
    {
        var prev=null, currMax=head(x), z=x;
        while( tail(z) )
        {
            ;;; z vísar á einhvern hala á listanum x,
```

```
            ;;; eins og hér er sýnt:
            ;;;
            ;;;  x=[x1,x2,...,xI,...,xN]
            ;;;            z=[xI,...,xN]
            ;;;
            ;;; currMax er stærsta gildið af x1,...,xI.
            ;;; Ef currMax er x1 þá er prev==null, annars
            ;;; vísar prev á næsta hlekk fyrir framan þann
            ;;; sem inniheldur currMax.
            ;;;
            if( head(tail(z))>currMax )
            {
                prev = z;
                currMax = head(tail(z));
            };
            z = tail(z);
        };
        if( !prev ) { return tail(x) };
        setTail(prev,tail(tail(prev)));
        x;
    };
}}
;

"test1.mexe" = main in
!
{{

;;; Notkun: y = max(x);
;;; Fyrir:  x=[x1,x2,...,xN] er listi
;;;         gilda sem samanburðaraðgerðin >
;;;         virkar fyrir.
;;; Eftir:  y er stærsta gildið í x miðað
;;;         við samanburðaraðgerðina >.
max =
    fun(x)
    {
        var mx = head(x);
        var z = x;
        for(;;)
        {
            ;;; z vísar á einhvern hala á listanum x,
            ;;; eins og hér er sýnt:
            ;;;
            ;;;  x=[x1,x2,...,xI,...,xN]
            ;;;            z=[xI,...,xN]
            ;;;
            ;;; mx er stærsta gildið af x1,...,xI.
            ;;;
            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.

TÖL304G
Forritunarmál
Verkefnablað 9

Snorri Agnarsson

21. október 2019

Ekkert verkefni

Við sleppum verkefnablaði 9 til að koma taktinum aftur í lag. Skilum á verkefnum 8 í Gradescope er seinkað til að fylla í skarðið.

TÖL304G
Forritunarmál
Verkefnablað 10

Snorri Agnarsson

27. október 2019

Verkefni

Í öllum verkefnunum skulu öll föll og boð (*messages*), 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.

Hópverkefni

Skrifið einingu fyrir hlutbundna forgangsbiðröð í Morpho. Athugið að verkefni vikunnar á undan var óhlutbundin forgangsbiðröð í Morpho þannig að hér er ekki um sama verkefni að ræða. Hins vegar mætti nota lausn frá fyrri viku sem hluta af smíð þessa verkefnis (og reyndar öfugt, ef þú hefur tímavél til umráða).

Útflutt úr einingunni skal vera eitt stef, sem er smiðurinn fyrir forgangsbiðröð. Smiðurinn þarf ekki taka að neitt viðfang en má taka eitt viðfang sem er þá hámarksfjöldi gilda í forgangsbiðröðinni. Innflutt skal vera samanburðaraðgerð, eins og í síðustu viku, en einnig öll boð fyrir forgangsbiðröðina, s.s. til að athuga hvort forgangsbiðröðin er tóm, til að bæta gildi í með tilteknum lykli og til að fjarlægja gildið sem hefur lykilinn með mesta forgangi og gila gildinu.

Öllum innfluttum og útfluttum atriðum skal lýsa í hönnunarskjali með notkunarlýsingu, forskilkyrði og eftirskilyrði. Útfærsla (smíð) skal innihalda skýra og nákvæma fastayrðingu gagna. Í þessu tilviki skal fastayrðing gagna lýsa innihaldi tilviksbreytna og sambandi þess innihalds við sameiginlega hugmynd okkar um forgangsbiðröð. Athugið að hafa upplýsingahuld í heiðri.

Einnig skal skrifa tvö prófunarforrit, sem nota forgangsbiðraðareininguna til að raða lista heiltalna, annars vegar í vaxandi röð og hins vegar í minnkandi röð. Sýna skal útkomur prófana. Leyfilegt er að sýna útkomurnar sem athugasemdir innan forritstexta viðkomandi prófunarforrita.

Einkunn fyrir hönnun gildir 40%, fyrir smíð einnig 40% og fyrir prófun 20%.

Einstaklingsverkefni

Skrifið einingu fyrir hlutbundna biðröð. Útflutt úr einingunni skal vera smiður sem tekur ekkert viðfang, innflutt verða boðin sem gera þær aðgerðir sem til þarf, þ.e.:

- Boð til að athuga hvort biðröðin er tóm.
- Boð til að bæta gildi aftast í biðröðina.
- Boð til að fjarlægja fremsta gildi úr biðröðinni.

Skila skal eftirfarandi (má allt vera í einni skrá):

- Hönnunarskjali með lýsingum útfluttra og innfluttra atriða (40%).
- Smíð með fastayrðingu gagna (40%).
- Prófunarforrit ásamt útskrift á útkomu prófunar (20%).

Sýna má útkomu prófunar sem athugasemd innan forritstexta.

TÖL304G
Forritunarmál
Verkefnablað 11

Snorri Agnarsson

3. nóvember 2019

Verkefni

Haskell verkefni:

Hópverkefni

1. Notið foldl eða foldr, sem eru innbyggð föll í Haskell, til að búa til Haskell fall sem tekur sem viðfang lista gilda [x1,x2,...,xN] og skilar listanum [[xN],...,[x2],[x1]]. Stofninn á fallinu skal vera eitt kall á foldl eða foldr með viðeigandi viðföngum.

2. Skrifið fall í Haskell sem tekur eitt viðfang sem er listi lista af fleytitölum og skilar summunni af margfeldunum af gildum innri listanna. Athugið að summa núll talna er 0.0 og margfeldi núll talna er 1.0.

Einstaklingsverkefni

1. Skrifið fall `listAll` í Haskell sem tekur viðföng `i`, `n` og `f` með `i<=n+1` og skilar listanum [f(i),f(i+1),...,f(n)]. Athugið að `listAll 1 0 f` skal skila [].

2. Skrifið Haskell fall `powerList` sem tekur lista [x1,x2,...,xN] sem viðfang og skilar lista af öllum undirlistum listans. Athugið, að eins og áður, þá telst listi [y1,y2,...,yK] vera undirlisti listans [x1,x2,...,xN] ef unnt er að fá [y1,y2,...,yK] með því að fjarlægja núll eða fleiri xI úr [x1,x2,...,xN] og breyta ekki röð gildanna sem eftir eru.

TÖL304G
Forritunarmál
Verkefnablað 12

Snorri Agnarsson

10. nóvember 2019

Verkefni

Hópverkefni

1. Hvers vegna hefði ekki verið skynsamlegt af hönnuðum Java að skilgreina að fjölnota biðraðaklasi Queue<A> virkaði þannig að boð f sem tekur viðfang af tagi Queue<A> megi í staðinn taka viðfang af tagi Queue<B>, þar sem B er *undirklasi* A?

   Sýnið forritskafla sem væri þá löglegur, hvað varðar tögun (*typing*), en óskynsamlegur, hvað varðar merkingu. Rökstyðjið að forritskaflinn sé óskynsamlegur.

   Reiknið með því að Queue<X> hafi boð `put` og `get` sem taka við og skila gildum af tagi X.

2. Hvers vegna hefði ekki verið skynsamlegt af hönnuðum Java að skilgreina að fjölnota biðraðaklasi Queue<A> virkaði þannig að boð f sem tekur viðfang af tagi Queue<A> megi í staðinn taka viðfang af tagi Queue<B>, þar sem B er *yfirklasi* A?

   Sýnið forritskafla sem væri þá löglegur, hvað varðar tögun (*typing*), en óskynsamlegur, hvað varðar merkingu. Rökstyðjið að forritskaflinn sé óskynsamlegur.

   Reiknið með því að Queue<X> hafi boð `put` og `get` sem taka við og skila gildum af tagi X.

3. Íhugið eftirfarandi þrjú fjölnota skil (*interface*) í Java.

```
// Hlutir af tagi Producer<T> eru framleiðendur
// (producer) á gildum af tagi T.
public interface Producer<T>
{
    // Notkun: T x = p.get();
    // Fyrir:  p vísar á framleiðanda gilda af
    //         tagi T.
    // Eftir:  x vísar á gildi sem p framleiddi.
    T get();
}

// Hlutir af tagi Consumer<T> eru neytendur
// (consumer) á gildum af tagi T.
public interface Consumer<T>
{
    // Notkun: c.put(x);
    // Fyrir:  x er gildi af tagi T,
    //         c er neytandi (consumer) gilda
    //         af tagi T.
    // Eftir:  c hefur neytt (consumed) x.
    put(T x);
}

// Hlutir af tagi Container<T> eru geymar
// (container) sem innihalda gildi af tagi T.
public interface Container<T>
{
    // Notkun: c.put(x);
    // Fyrir:  x er gildi af tagi T,
    //         c er geymir (container) gilda
    //         af tagi T.
    // Eftir:  x hefur verið sett í c.
    put(T x);

    // Notkun: T x = c.get();
    // Fyrir:  c vísar á geymi gilda af tagi T.
    // Eftir:  x vísar á gildi sem var fjarlægt
    //         úr c.
    T get();
}
```

Er rökrétt að líta svo á að

a) `Container<String>` sé undirtag skilanna `Container<Object>`?

b) `Container<Object>` sé undirtag skilanna `Container<String>`?

c) `Consumer<String>` sé undirtag skilanna `Consumer<Object>`?

d) `Consumer<Object>` sé undirtag skilanna `Consumer<String>`?

e) `Producer<String>` sé undirtag skilanna `Producer<Object>`?

f) `Producer<Object>` sé undirtag skilanna `Producer<String>`?

Við lítum svo á að rökrétt sé að klasi eða skil B séu undirtag klasa eða skila A ef allir hlutir af tagi B uppfylla samninginn sem klasi eða skil A skilgreina fyrir sína hluti. Aðeins þarf að svara „já“ eða „nei“ í hverju tilviki.

Athugið að ekki er verið að spyrja hvort Java forritunarmálið lítur svo á að viðkomandi vensl gildi milli klasanna heldur hvort venslin séu rökrétt, þ.e. hvort þau séu í samræmi við eðlilegar röksemdafærslureglur í forritun.

Einstaklingsverkefni

Skrifið klasa `V12pq` fyrir fjölnota forgangsbiðröð í Jövu. Unnt skal vera að nota forgangsbiðröðina auðveldlega í venjulegum kringumstæðum án þess að þurfa að nota tagvarpanir (*casts*) til að koma gildum í forgangsbiðröðina og úr henni. Þegar gildum er bætt í forgangsbiðröðina skal tiltaka tvö gildi, k og v, þar sem k er lykill og v er gildi. Þið skuluð leyfa að tagið á v sé hvaða hluttag sem er (en ekki frumstæð tög eins og `int` eða `double`, það gengur ekki í Java). Tagið á k þarf hins vegar að vera skorðað þannig að sérhver tvö gildi af því tagi séu samanburðarhæf, svipað eins og í MyArray klasanum. Aðgerðin sem bætir við forgangsbiðröðina skal þá geyma gildið v undir lyklinum k, sem þá skilgreinir forgang gildisins v í forgangsbiðröðinni.

Þið megið nota þá klasa sem til eru í Java til að útfæra ykkar fjölnota forgangsbiðröð. Til dæmis getur verið gagnlegt að nota klasann TreeMap[1]. Sá klasi er útfærsla á rauð-svörtum trjám. Eins og við munum frá Tölvunarfræði 2 getum við með slíkum trjám gert ýmsar gagnlegar aðgerðir með tímaflækju $O(\log n)$. Munið samt að leyfa að tvö gildi í forgangsbiðröð hafi sama gildi og að par (k, v) sé margfalt inni í forgangsbiðröðinni. **Einföld notkun á `TreeMap<K,V>` mun því ekki duga til að útfæra forgangsbiðröðina** því `TreeMap<K,V>` leyfir aðeins eitt gildi með tilteknum lykli. Þið megið, ef þið viljið, nota klasana `V12Bag`, `V12Pair` og `V12ComparablePair`, sem finna má í Uglunni. Annar kostur er að nota klasann `java.util.LinkedList`. Í öllum tilvikum er mælt með að nota `java.util.TreeMap`. Einnig getur verið gagnlegt að nota klasann `java.math.BigInteger`.

Skilið í Gradescope forritstextanum fyrir V12pq, þ.e. V12pq.java. Munið að vanda til innfellinga og munið að vanda lýsingar á öllum boðum í klasanum (þ.e. Notkun/-

[1] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/TreeMap.html

Fyrir/Eftir) og munið að skrifa skýra og rétta fastayrðingu gagna inni í forritstextanum. Munið einnig eftir meginreglunni um upplýsingahuld. Ekki skila forritstexta fyrir aðra klasa sem þið skrifuðuð ekki sjálf.

Í Uglunni er beinagrind fyrir skrána V12pq.java.

Einkunnin fyrir verkefnið er 50% fyrir hönnun (þ.e. fyrir Notkun/Fyrir/Eftir), 50% fyrir smíð, þ.m.t. fastayrðing gagna.

TÖL304G
Forritunarmál
Verkefnablað 13

Snorri Agnarsson

17. nóvember 2019

Verkefni

Hópverkefni

Skrifið Java forrit þar sem einn þráður framleiðir heiltölurnar 1, 2, ..., 10000, sendir þær hverja eftir aðra í annan þráð, sem leggur þær allar saman og skrifa loks summuna þegar 10000 tölur hafa verið mótteknar. Inni í forritinu skuluð þið hafa þrjá klasa, sem eiga að vera innri klasar í ykkar aðalklasa, og skulu heita Producer, Container og Consumer. Producer og Consumer eiga að vera þræðir[1], en Container á að vera ílát sem getur annað hvort verið tómt eða innihaldið eitt gildi.

Container klasinn á að vera einhvern veginn svona:

```
// Hvert tilvik af þessum klasa er ílát fyrir
// heiltölu sem þræðir geta samtímis unnið
// með.
static class Container
{
  boolean isEmpty = true;
  int theValue;
  // Fastayrðing gagna:
  //   Ílátið er tómt þþaa isEmpty sé true.
  //   Ef ílátið er ekki tómt þá inniheldur
  //   theValue gildið í ílátinu.
```

[1] Slíka þræði má búa til annaðhvort með því að smíða `Thread` hluti úr frá tveimur mismunandi `Runnable` hlutum eða með því að búa til tvo mismunandi undirklasa klasans `Thread`.

```
  // Notkun: c.put(x);
  // Eftir:  Búið er að setja x í ílátið c.
  //         Ef til vill þurfti að bíða eftir
  //         að ílátið tæmdist áður en það
  //         tókst.
  public synchronized void put( int x )
    throws InterruptedException
  {
    while( !isEmpty ) wait();
    isEmpty = false;
    theValue = x;
    notifyAll();
  }

  // Notkun: x = c.get();
  // Eftir:  Búið er að sækja x úr ílátinu c.
  //         Ef til vill þurfti að bíða eftir
  //         að ílátið fylltist áður en það
  //         tókst.
  public synchronized int get()
    throws InterruptedException
  {
    while( isEmpty ) wait();
    int x = theValue;
    isEmpty = true;
    notifyAll();
    return x;
  }
}
```

Athugið notkunina á boðunum wait() og notifyAll() og virkni þeirra í samhengi við lykilorðið synchronized. Þessi tvö boð eru skilgreind í klasanum Object og eru því til staðar í sérhverjum Java hlut.

Einstaklingsverkefni

Skrifið Morpho forrit eða fall sem gerir sama og Java forritið að ofan. Þið megið skrifa forritið sem Morpho script fall eða þýðingarhæft forrit eftir því hvað ykkur hentar. Einfaldast er að nota channel sem samskiptarás, sem virkar þá svipað og Container að ofan. Fallið makeChannel býr til nýjan channel. Til að senda gildi inn í channel má nota *tvíundaraðgerðina* <−. Til að sækja gildi úr channel má nota *einundaraðgerðina*

`<-`. Þessar aðgerðir hafa sama nafn en eru ekki sama aðgerðin. Ef breytan `c` inniheldur channel þá mun segðin `c<-x` senda (skrifa) gildið `x` í þann channel. Segðin `x = <-c` mun sækja (lesa) gildi úr þeim channel og setja í breytuna `x`.