TÖL304G — Forritunarmál

Haustpróf 13. desember 2011.

Engin hjálpargögn eru leyfileg.

Öll dæmi gilda jafnt.

Munið að skrifa notkunarlýsingu með forskilyrði og eftirskilyrði fyrir sérhvert stef og fastayrðingu gagna fyrir sérhverja útfærslu gagnamóts.

Athugið vel: Svara þarf tilskildum fjölda dæma úr hverjum hluta prófsins. Að því skilyrði uppfylltu gilda **12 bestu dæmi** til einkunnar. Þið hafið því 15 mínútur fyrir hvert dæmi að meðaltali. Byrjið því á að svara dæmum sem krefjast stuttra svara og þið getið auðveldlega svarað.

Hluti I: Bálkmótun o.fl.

Svarið a.m.k. 3 dæmum úr þessum hluta

1. Lýsið því hvernig vakningarfærslur í bálkmótuðum forritunarmálum eru tengdar saman með stýrihlekkjum (*control link, dynamic link*) og aðgangshlekkjum (þ.e. tengihlekkjum, *access link*) og hvernig hlekkir eru notaðir til að gefa aðgang að breytum í mismunandi földunarhæðum.

2. Gerið ráð fyrir földun í bálkmótuðu forritunarmáli eins og myndin að neðan sýnir. Forritstextinn sýnir dæmi um slíka földun í Scheme.

   Hverjar eftirfarandi fullyrðinga eru þá sannar? Eitt rangt svar leiðir til þess að ekkert stig fæst.

   (a) Kalla má á S úr P.

   (b) Kalla má á S úr Q.

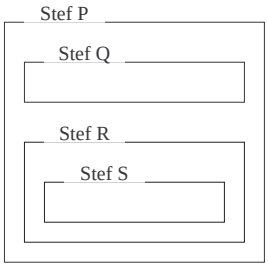
   (c) Kalla má á S úr R.

   (d) Kalla má á P úr S.

   (e) Kalla má á Q úr S.

   (f) Í S má nota staðværar breytur úr P.

   (g) Í R má nota staðværar breytur úr Q.




```
(define (P ...)
  (define (Q ...)
    ...
  )
  (define (R ...)
    (define (S ...)
      ...
    )
    ...
  )
  ...
)
```

3. (a) Hvaða upplýsingar eru geymdar í vakningarfærslum forritunarmála?
   (b) Hverjar þessara upplýsinga eru aðeins í bálkmótuðum forritunarmálum og hvers vegna?
   (c) Nefnið tvö forritunarmál þar sem vakningarfærslur eru yfirleitt geymdar á hlaðanum og tvö þar sem vakningarfærslur eru yfirleitt geymdar í kös. Lýsið einhverjum veigamiklum afleiðingum þeirrar ákvörðunar að geyma vakningarfærslurnar annars vegar í kös og hins vegar á hlaða á það hvers konar virkni forritunarmálin geta auðveldlega boðið upp á.

4. Hverjar eftirfarandi fullyrðinga um lokanir (*closure*) eru sannar?
   (a) Lokanir innihalda aðgangshlekk (tengihlekk, *access link*, *static link*).
   (b) Lokanir innihalda stýrihlekk (*control link*, *dynamic link*).
   (c) Lokanir innihalda vakningarfærslur.
   (d) Lokanir innihalda bendi á vakningarfærslur.

(e) Lokanir innihalda bendi á stef eða fall.

(f) Ástæða þess að lokanir eru sveigjanlegri í notkun í Scheme en í Pascal er að í Scheme má geyma lokanir í kös.

(g) Ástæða þess að lokanir eru sveigjanlegri í notkun í Scheme en í Pascal er að í Scheme má geyma vakningarfærslur í kös.

(h) Í Scheme má lokun vera skilagildi úr falli, en ekki í Standard Pascal.

(i) CAML hefur lokanir.

(j) Haskell hefur lokanir.

(k) C hefur lokanir.

(l) C++ hefur lokanir.

(m) Morpho hefur lokanir.

(n) Java hefur lokanir.

5. Svarið báðum eftirfarandi:

   - Eftirfarandi forritstexti er í einhverju ímynduðu forritunarmáli. (Spyrjið ef ykkur finnst merking eða málfræði ekki augljós.)

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

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

     Hvað skrifar þetta forrit (fjögur gildi í hvert skipti) ef viðföngin eru

     – gildisviðföng?
     – tilvísunarviðföng?
     – nafnviðföng?

   - Gerið grein fyrir öllum mögulegum viðfangaflutningum í eftirfarandi forritunarmálum: C++, Java, Haskell, Scheme, Morpho.

6. Í eftirfarandi forritstexta eru lokanir (meðal annars) búnar til og notaðar. Hverjar af breytunum a,. . . ,g innihalda lokanir? Hvaða gildi innihalda hinar breyturnar? Hvaða segðir valda villu í keyrslu?

```
(define a 1)
(define b (lambda (x) (+ a x)))
(define (c x) (+ x a))
(define (d x) (lambda (y) (+ x y)))
(define e (d 10))
(define f (e 20))
(define g (f 30))
```

Hluti II: Listavinnsla o.fl.

Svarið a.m.k. 3 dæmum úr þessum hluta

7. Látum gildi af gerð G vera þau gildi sem uppfylla eftirfarandi lýsingu:

   - Heiltölur eru af gerð G.
   - Ef $x_1, \ldots, x_n$ eru af gerð G þá er listinn $[x_1, \ldots, x_n]$ (eða $(x_1 \ldots x_n)$ í Scheme) af gerð G. ($n \geq 0$).
   - Engin önnur gildi eru af gerð G.

   Skrifið fall í Scheme eða Morpho sem skilar summu heiltalnanna í gildi af gerð G.

   Athugið að í Scheme skilar `(ispair? x)` sönnu gildi þþaa `x` vísi á par og í Morpho gerir `isPair(x)` það sama. Í Scheme skilar `(null? x)` sönnu þþaa `x` sé tómi listinn og í Morpho gerir `x==[]` það sama.

8. Skrifið fall í Scheme sem tekur tvö viðföng, `x` og `y`, og skilar falli (lokun) `f` sem er þeirri náttúru gætt að `(f 0)` er `x` og `(f 1)` er `y`.

   Gerið grein fyrir hvort hægt er að skrifa svona fall á svipaðan hátt í eftirfarandi forritunarmálum: C++, Java, Morpho, CAML, Haskell. Rökstyðið.

9. Skrifið fall í Scheme, Haskell, CAML eða Morpho, sem tekur eina heiltölu $n \geq 0$ sem viðfang og skilar listanum ($n\ n-1 \ldots 1$).

10. Skrifið fall í Scheme, CAML eða Morpho sem tekur eitt viðfang, sem skal vera listi fleytitalna $[x_1, \ldots, x_n]$ (ekki tómur). Fallið skal skila fleytitölunni $(\ldots((x_1 \ominus x_2) \ominus x_3) \ominus \ldots) \ominus x_n$. Þar sem tvíundaraðgerðin $x \ominus y$ er skilgreint sem $x/(1+y)$. Nota má föllin `hd` og `tl` í CAML, en ekki `it_list` eða `list_it`.

11. Eftirfarandi er lögleg Haskell skilgreining. Hvað er að skilgreiningunni? Sýnið hvernig laga má skilgreininguna, annars vegar í listarithætti og hins vegar með einhverri annarri aðferð, t.d. með do-rithætti. Hraði lausnarinnar skiptir ekki máli.

```
pyth = [(x,y,z)|x<-[1..],y<-[1..],z<-[1..],x^2+y^2==z^2]
```

Hluti III: Einingaforritun o.fl.

Svarið a.m.k. 3 dæmum úr þessum hluta

12. Skrifið hönnunarhluta einingar í Morpho, Jövu **eða** C++ fyrir fjölnota forgangsbiðröð. Þið ráðið hvort forgangsbiðraðirnar eru hlutbundnar.

13. Skrifið skil (*interface*) í Jövu fyrir biðröð heiltalna (athugið að biðröð er alls ekki það sama og forgangsbiðröð).

14. Skrifið fjölnota klasa í C++ fyrir hlaða. Pakkinn skal vera fjölnota (*template*) og gefa kost á notum fyrir gildi af sem fjölbreytilegustu tagi. Tilgreinið hvaða skilyrði tag þeirra gilda sem geymd eru á hlaðanum þarf að uppfylla til að unnt sé að búa til hlaða gilda af því tagi.

15. Skrifið hönnunarskýrslu fyrir fjölnota pakka eða einingu í C++ **eða** Morpho sem býður upp á reikninga með tvinntölur. Ekki skal forrita eininguna, en ef skilað er C++ lausn skal allur skilgreiningarhluti pakkans vera með (header skrá). Unnt skal vera að nota eininguna til að skilgreina tvinntölur þar sem raunhlutinn er hvers konar kommutala, hvernig svo sem kommutölurnar eru smíðaðar, hvort sem þær eru innbyggðar í málið eða smíðaðar af notanda á einhvern hátt.

    Munið að gera grein fyrir öllum útfluttum *og innfluttum* atriðum með notkunarlýsingu hvers atriðis.

16. Skrifið hönnunarskýrslu fyrir fjölnota einingu í Morpho fyrir reikninga með ræðar tölur. Aðgerðir á heiltölur skulu vera innfluttar og ekki skal gefa sér neinar forsendur um það hvernig heiltölurnar eru geymdar.

17. Hvert þarf sambandið milli forskilyrða og eftirskilyrða í undirklösum og yfirklösum að vera til að röksemdafærsla sé traust?

18. Skrifið einingu í Morpho fyrir hlaða. Bæði skal skrifa hönnun og smíð. Gleymið ekki fastayrðingu gagna. Þið ráðið hvort einingin er hlutbundin.

Hluti IV: Blandað efni

Ekki þarf endilega að svara neinu dæmi úr þessum hluta, en ekki gleyma að svara 12 dæmum í heild.

19. Gerið grein fyrir þremur af eftirfarandi aðferðum til viðfangaflutninga:

- Gildisviðföngum
- Tilvísunarviðföngum
- Afritsviðföngum
- Nafnviðföngum eða lötum viðföngum

Fyrir hverja aðferð skuluð þið einnig nefna eitt forritunarmál sem styður aðferðina.

20. Nefnið þrjár ruslasöfnunaraðferðir og lýsið einni þeirra. Lýsið einnig kostum hennar og göllum, sérstaklega með tilliti til minnisnotkunar og tímaflækju.

21. Svarið báðum eftirfarandi:

    - Gefin er eftirfarandi EBNF mállýsing:

      ```
      Exp = 't'
          | '(', Exp, ')'
          | Exp, '+', Exp
          | Exp, '*'
          | '-', Exp
          ;
      ```

      Hverjir eftirfarandi strengja eru í málinu (tvö röng svör gefa núll):

      (a) (t+2)
      (b) t**
      (c) t+
      (d) t+t*
      (e) (t*)+t
      (f) –t**
      (g) t-t*
      (h) (-t)-t

    - Sýnið BNF, EBNF eða málrit fyrir mál löglegra formúlna með svigum, tölunni 1 og tvíundaraðgerðinni $+$. Dæmi um löglegar formúlur eru þá t.d. 1, (1) og $1 + 1$, en ekki t.d. $11 + 1$ (því 11 er þá ekki lögleg tala).

22. Svarið báðum eftirfarandi:

    - Teiknið endanlega stöðuvél (löggenga eða brigðgenga) fyrir málið yfir stafrófið $\{a, b\}$ sem inniheldur þá strengi þar sem mismunurinn á fjölda a-a og b-a er oddatala.

- Skrifið reglulega segð fyrir mál þeirra strengja yfir stafrófið $\{a, b\}$ sem innihalda strenginn $abbabab$ sem hlutstreng.

TÖL304
Forritunarmál
Próf 2011
Öll dæmi gilda jafnt

Hluti I: Bálkmótun o.fl.

Kamilla svaraði

1. Lýsið því hvernig vakningarfærslur í bálkmótuðum forritunarmálum eru tengdar saman með stýrihlekkjum (control link, dynamic link) og aðgangshlekkjum (þ.e. tengihlekkjum, access link) og hvernig hlekkir eru notaðir til að gefa aðgang að breytum í mismunandi földunarhæðum.

Stýrihlekkurinn bendir ávallt á vakningarfærslu þess stefs sem kallaði. Í forritunarmálum sem styðja halaendurkvæmni, þá bendir stýrihlekkurinn á vakningarfærslu þess stefs sem á að fá niðurstöðuna úr núverandi kalli.

Tengihlekkurinn bendir ávallt á vakningarfærslu þess stefs sem inniheldur viðkomandi stef textalega séð (lexically enclosing scope). Viðeigandi vakningarfærsla er ávallt sú vakningarfærsla sem inniheldur breyturnar í næstu földunarhæð og er næstum alltaf nýjasta vakningarfærsla ytra stefsins.

Tengihlekkurinn vísar á vakningarfærslu sem inniheldur breyturnar í næstu földunarhæð fyrir ofan og síðan koll af kolli. Tengihlekkirnir mynda keðju og það er hægt að ganga hana til að komast í breytur í mismunandi földunarhæðum.

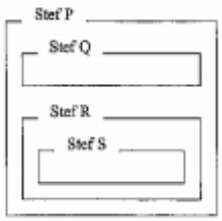
Kamilla svaraði

2. Gerið ráð fyrir földun í bálkmótuðu forritunarmáli eins og myndin að neðan sýnir. Forritstextinn sýnir dæmi um slíka földun í Scheme.

Hverjar eftirfarandi fullyrðinga eru þá sannar? Eitt rangt svar leiðir til þess að ekkert stig fæst.

a) Kalla má á S úr P - **Nei**
b) Kalla má á S úr Q - **Nei**
c) Kalla má á S úr R - **Já**
d) Kalla má á P úr S - **Já**
e) Kalla má á Q úr S - **Já**
f) Í S má nota staðværar breytur úr P - **Já**
g) Í R má nota staðværar breytur úr Q - **Nei, því Q og R eru óskyld föll**

```
(define (P ...)
  (define (Q ...)
    ...
  )
  (define (R ...)
    (define (S ...)
      ...
    )
    ...
  )
  ...
)
```



Dagmar svaraði

3. a) Hvaða upplýsingar eru geymdar í vakningarfærslum forritunarmála?
b) Hverjar þessara upplýsinga eru aðeins í bálkmótuðum forritunarmálum og hvers vegna?
c) Nefnið tvö forritunarmál þar sem vakningafærslur eru yfirleitt geymdar á hlaðanum og tvö þar sem vakningarfærslur eru yfirleitt geymdar í kös. Lýsið einhverjum veigamiklum afleiðingum þeirrar ákvörðunar að geyma vakningarfærslurnar annars vegar í kös og hins vegar á hlaða á það hvers konar virkni forritunarmálin geta auðveldlega boðið upp á.

Svar: a) viðföng, staðværar breytur, vendivistfang, stýrihlekkur og tengihlekkur.
b) Erum bara með tengihlekk þar sem við erum með föll inní föllum og þurfum að geta vísað í þau.
c) C++ og Java geymdar á hlaða. Ef í kös geturðu skilað lokunum sem gildi úr falli, fall getur skilað falli.

Ekki hægt þegar við erum með á hlaða. og dæmi um forritunar mál eru þá scheme, haskell og þessi bálkmótuðu mál.

Dagmar svaraði

4. Hverjar eftirfarandi fullyrðinga um lokanir (closure) eru sannar?
a) Lokanir innihalda aðgangshlekk (tengihlekk, access link, static link) **Satt**
b) Lokanir innihalda stýrihlekk (control link, dynamic link) **Ósatt**
c) Lokanir innihalda vakningarfærslur **Ósatt (lokanir innihalda tilvísanir (bendi) á vakningafærslur, en innihalda ekki vakningafærslur)**
d) Lokanir innihalda bendi á vakningarfærslur **Satt**
e) Lokanir innihalda bendi á stef eða fall **Satt**
f) Ástæða þess að lokanir eru sveigjanlegri í notkun í Scheme en í Pascal er að í Scheme má geyma lokanir í kös **Ósatt (ekki lokanir í kös, vakningafærslur í kös)**
g) Ástæða þess að lokanir eru sveigjanlegri í notkun í Scheme en í Pascal er að í Scheme má geyma vakningarfærslur í kös **Satt**
h) Í Scheme má lokun vera skilagildi úr falli, en ekki í Standard Pascal **Satt**
i) CAML hefur lokanir **Satt**
j) Haskell hefur lokanir **Satt**
k) C hefur lokanir **Ósatt**
l) C++ hefur lokanir **Ósatt**
m) Morpho hefur lokanir **Satt**
n) Java hefur lokanir **Ósatt**

Hrefna svaraði - leiðrétt af Kamillu

5. Svarið báðum eftirfarandi:

- Eftirfarandi forritstexti er í einhverju ímynduðu forritunarmáli. (Spyrjið ef ykkur finnst merking eða málfræði ekki augljós

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

  Hvað skrifar þetta forrit (fjögur gildi í hvert skipti) ef viðföngin eru:
  - gildisviðföng - call by value?
    2 0
    0 2
  - tilvísunarviðföng - call by reference?
    2 2
    1 2
  - nafnviðföng - call by name?
    2 4
    2 2
- Gerið grein fyrir öllum mögulegum viðfangaflutningum í eftirfarandi forritunarmálum: C++, Java, Haskell, Scheme, Morpho.

Java, Scheme og Morpho eru aðeins með gildisviðföng, en C++ er bæði með gildis- og tilvísunarviðföng. Haskell er bara með löt viðföng.

Kamilla svaraði

6. Í eftirfarandi forritstexta eru lokanir (meðal annars) búnar til og notaðar. Hverjar af breytunum a,....,g innihalda lokanir? Hvaða gildi innihalda hinar breyturnar? Hvaða segðir valda villu í keyrslu?

(define a 1) - talan 1 er ekki lokun, lokun er eitthvað fall sem hægt er að kalla á í Scheme
(define b (lambda (x) (+ a x))) - lokun
(define (c x) (+ x a)) - lokun
(define (d x) (lambda (y) (+ x y))) - lokun og d af einhverju er líka lokun
(define e (d 10)) - lokun, skilar lambda dótinu sem skilar lokun
(define f (e 20)) - f er ekki lokun (f er ekki fall) f skilar 30
(define g (f 30)) - veldur villu því f er ekki fall, reynum að kalla á 30 beitt á 30

Hluti II: Listavinnsla o.fl.

Valborg svaraði

7. Látum gildi af gerð G vera þau gildi sem uppfylla eftirfarandi lýsingu:

- Heiltölur eru af gerð G
- Ef $x_1,....,x_n$ eru af gerð G þá er listinn $[x_1,...,x_n]$ (eða $(x_1...x_n)$ í Scheme) af gerð G. (n>= 0)
- Engin önnur gildi eru af gerð G

Skrifið fall í Scheme eða Morpho sem skilar summu heiltalnanna í gildi af gerð G. Athugið að í Scheme skilar (ispair? x) sönnu gildi þþaa x vísi á par og í Morpho gerir isPair(x) það sama. Í Scheme skilar (null? x) sönnu þþaa x sé tómi listinn og í Morpho gerir x==[] það sama.

Í Scheme

```
;Notkun: sum G
;Fyrir: G uppfyllir gefin skilyrði fyrir G (að ofan)
```

- Heiltölur eru af gerð G
- Ef x1,....,xn eru af gerð G þá er listinn [x1,...,xn] (eða (x1...xn) í Scheme) af gerð G. (n>= 0)
- Engin önnur gildi eru af gerð G

```
;Gildi: Ef G er listi talna er gildið summa talnanna en ef G er stök tala er gildið G.
 (define (sum G)
   ;Notkun (summa x u)
   ;Fyrir: x er af tagi G og u er heiltala.
   ;Gildi: u = x ef x er heiltala, u = x1 + x2 + … + xN ef x er listi heiltalna.
   (define (summa x u)
         (if (null? x)
                u
               (summa (cdr x) (+ u (car x)))
         )
   )
         (if (ispair? G)
         (summa G 0)
         G
   )
)
```

ATH - Snorri svaraði dæmi 7 svona í tíma þegar við fórum yfir prófið

```
(define (sumG x)
        (if (isPair? x)
                (+ (sumG (car x))
                        (sumG (cdr x))
                )
                (if (null? x)
                        0
```

```
                x
            )
        )
)
```

Hrefna svaraði

8. Skrifið fall í Scheme sem tekur tvö viðföng, x og y, og skilar falli (lokun) f sem er þeirri náttúru gætt að (f 0) er x og (f 1) er y. Z verður að vera boole gildi.
Gerið grein fyrir hvort hægt er að skrifa svona fall á svipaðan hátt í eftirfarandi forritunarmálum: C++, Java, Morpho, CAML, Haskell. Rökstyðjið.

```
define (f x y) (lambda(z)(if(=z 0)x y)))
```

Ekki er hægt að skrifa svona fall í Java eða C++, því þetta byggir á bálkmótun og að hægt sé að skila lokun (væri hægt að búa til hlut sem gerir svipað en ekki fall).
Væri hægt í Morpho, CAML og Haskell því þau styðja lokanir.

Hrefna og Kamilla svöruðu

9. Skrifið fall í Scheme, Haskell, CAML eða Morpho sem tekur eina heiltölu n>=0 sem viðfang og skilar listanum (n n-1 ...1).
Sjá lausn við dæmi 7 á miðannaprófi fyrir CAML og Morpho, ef þið rúllið svona.

í haskell:

```
-- Notkun:      rindex n
-- Fyrir:n er heiltala n >= 0
-- Eftir: Listinn [n, n-1,...,1]
rindex n = reverse[1 … n]
```

Í Scheme:
Notkun: index n
Fyrir: n er heiltala n >= 0
Gildi: Listinn (n n-1...2 1)

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

Önnur útgáfa (snúum listanum við í lokin með reverse)

```
(define (index n)
        (define h n x)
                (if (= n 0)
                        x
                        (h (- n 1) (cons n x))
                )
        )
        (h n '())
)
reverse x
```

Þórunn svaraði en samt ekki NEW AND IMPROVED!!! LIKE YOU'VE NEVER SEEN BEFORE

10. Skrifið fall í Scheme, CAML eða Morpho sem tekur eitt viðfang, sem skal vera listi fleytitalna [x_1,...,x_n] (ekki tómur). Fallið skal skila fleytitölunni (...(($x_1 \ominus x_2$)$\ominus x_3$)$\ominus$...)$\ominus x_n$. Þar sem tvíundaraðgerðin $x \ominus y$ er skilgreind sem x/(1+y). Nota má föllin hd og tl í CAML, en ekki it_list eða list_it.

Lausn af miðannarprófi 2012 við svipaðri spurningu, næ kannski að breyta henni í rétt, dæmið var

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

```
;; Notkun: (insert f u x)
;; Fyrir:  f er tvíundarfall, u er fleytitala, x er listi fleytitalna
;; Gildi:  z=(f ... (f (f (x1 x2) x3) ...) xN)
(define (insert f u x)

 (if (null? x)
     u
    (insert f (f u (car x)) (cdr x))
  )
)
(define (gamangaman x)
  (insert (lambda (x,y)(/x (+ 1 y)))(car x)(cdr x)
)
```

Anna svaraði - Hvað með do rithátinn?

11. Eftirfarandi er lögleg Haskell skilgreining. Hvað er að skilgreiningunni? Sýnið hvernig laga má skilgreininguna, annars vegar í listarithætti og hins vegar með einhverri annarri aðferð, t.d. með do-rithætti. Hraði lausnarinnar skiptir ekki máli.

```
pyth = [(x,y,z)|x<-[1..], y<-[1..],z<-[1..], x^2+y^2==z^2]
```

Svar:

Þetta er óendanleg lykkja. Það eru engin efri mörk í y og z listanum og þess vegna er haldið endalaust áfram að leita eftir stökum sem uppfylla x^2+y^2==z^2.

Laga má skilgeininguna með því að gera:

```
pyth = [(x,y,z) | z <-[1..], x<-[1..z], y<-[x..z], x^2+y^2==z^2]
```

eða:

```
pyth = do
        z <- [1..]
        x <- [1.z]
        y <- [x..z]
        if x^2+y^2==z^2 then [(x,y,z)] else []
```

Hluti III: Einingaforritun o.fl.

Hrefna svaraði

12. Skrifið hönnunarhluta einingar í Morpho, Jövu eða C++ fyrir fjölnota forgangsbiðröð. Þið ráðið hvort forgangsbiðraðirnar eru hlutbundnar.

Morpho (af fl9.pdf):

{;;;

Hönnun

======

Útflutt

Notkun: pq = makePriQueue();
Eftir: pq er ný tóm forgangsbiðröð.
Innflutt

Notkun: pq.add(x);
Fyrir: pq er forgangsbiðröð.
x er gildi af þeirri gerð sem við leyfum að geymd séu í forgangsbiðröðinni (sbr. lýsingu á <<==).
Eftir: Búið er að bæta x í pq.

Notkun: b = pq.isEmpty();
Fyrir: pq er forgangsbiðröð.
Eftir: b er satt ef pq er tóm, annars ósatt.

Notkun: x = pq.getFirst();
Fyrir: pq er forgangsbiðröð, ekki tóm.
Eftir: x er það gildi úr pq sem var fremst gilanda í pq.
x hefur verið fjarlægt úr pq.

Notkun: b = x <<== y;
Fyrir: x og y eru gildi af þeirri gerð sem við leyfum að geymd séu í forgangsbiðröð.
Eftir: b er satt ef x verður að vera á undan y, ósatt ef y verður að vera á undan x og getur verið annað hvort satt eða ósatt annars (ef x og y mega vera í hvaða röð sem er)
;;;}

Dæmi um PriQueue í Java: http://t2.hhg.to/PriQueue.java

Hrefna svaraði Dagmar breytti.
13. Skrifið skil (interface) í Jövu fyrir biðröð heiltalna (athugið að biðröð er alls ekki það sama og forgangsbiðröð).

Stal þessum kóða til að nota í interface: http://t2.hhg.to/Queue.java Önnur lausn á fl13.pdf
Það á pottþétt að skrifa notkunarlýsingar líka, fyrst þetta er Snorri.

```
public interface Queue<T extends Comparable<? superT>>  //T er gildið á stökunum
{
        //global variables
        private node first:
        private node last;
        private int n:

        //functions
        private class Node { }  //Ekki nauðsynlegt, gerir í raun ekkert.
                //Finnur næstu nóðu (næsta gildi í eintengda listanum)
        public Queue() { }
                //Býr til biðröðina
        public boolean empty () { }  //Ekki nauðsynlegt
                //Er biðröðin tóm?
        public int size() { }  //Ekki nauðsynlegt
                //Hvað er biðröðin stór?
```

```
        public T peek() { }  //Ekki nauðsynlegt
                //Skoða næsta gildi, en fjarlægir það ekki.
        public void enqueue(T value) { }
                //Bætir við staki í biðröðina
        public T dequeue() { }
                //Tekur stak úr biðröðnni
}
```

Vantar

14. Skrifið fjölnota klasa í C++ fyrir hlaða. Pakkinn skal vera fjölnota (*template*) og gefa kost á notum fyrir gildi af sem fjölbreytilegustu tagi. Tilgreinið hvaða skilyrði tag þeirra gildi af sem geymd eru á hlaðanum þarf að uppfylla til að unnt sé að búa til hlaða gilda af því tagi.

Þórunn svarar:

15. Skrifið hönnunarskýrslu fyrir fjölnota pakka eða einingu í C++ eða Morpho sem býður upp á reikninga með tvinntölur. Ekki skal forrita eininguna, en ef skilað er C++ lausn skal allur skilgreiningahluti pakkans vera með (header skrá). Unnt skal vera að nota eininguna til að skilgreina tvinntölur þar sem raunhlutinn er hvers konar kommutala, hvernig svo sem kommutölurnar eru smíðaðar, hvort sem þær eru innbyggðar í málið eða smíðaðar af notanda á einhver hátt.

```
  ;;; Hönnun
  ;;;
  ;;; Útflutt:
  ;;;
;;; Notkun: c = makeComplex(r,i);
;;; Fyrir: r og i eru rauntölur
;;; Eftir: c er ný tvinntala með raunhluta r ;;; og þverhluta i.
;;;
;;; Innflutt:
;;;
;;; Notkun: r=c.x
;;; Fyrir: c er tvinntala
;;; Eftir: r er raunhluti c
;;;
;;; Notkun: i=c.i
;;; Fyrir: c er tvinntala
;;; Eftir: i er þverhluti c
;;;
;;; Notkun: z=a.+b
;;; Fyrir: a og b eru tvinntölur
;;; Eftir: z er ný tvinntala, z = a+b
;;;
;;; Notkun: z=a.-b
;;; Fyrir: a og b eru tvinntölur
;;; Eftir: z er ný tvinntala, z = a-b
;;;
;;; Notkun: z=a.*b
;;; Fyrir: a og b eru tvinntölur
;;; Eftir: z er ný tvinntala, z = a*b
;;;
;;; Notkun: z=a./b
;;; Fyrir: a og b eru tvinntölur
;;; Eftir: z er ný tvinntala, z = a/b
;;;
;;; Notkun: x=a+b
;;; Fyrir: a og b eru rauntölur
```

```
;;; Eftir: x er ný rauntala, x = a+b
;;;
;;; Notkun: x=a-b
;;; Fyrir: a og b eru rauntölur
;;; Eftir: x er ný rauntala, x = a-b
;;;
;;; Notkun: x=a+b
;;; Fyrir: a og b eru rauntölur
;;; Eftir: x er ný rauntala, x = a*b
;;;
;;; Notkun: x=a+b
;;; Fyrir: a og b eru rauntölur
;;; Eftir: x er ný rauntala, x = a/b
```

Valborg vill meina að þetta sé svarið:

16. Skrifið hönnunarskýrslu fyrir fjölnota einingu í Morpho fyrir reikninga með ræðar tölur. Aðgerðir á heiltölur skulu vera innfluttar og ekki skal gefa sér neinar forsendur um það hvernig heiltölurnar eru geymdar.

```
  ;;; Hönnun
  ;;;
  ;;; Útflutt:
  ;;;
;;; Notkun: c = raedtala(p,q);
;;; Fyrir: p og q eru heilar tölur
;;; Eftir: c er ný ræðtala með teljara p og nefnara q
;;;
;;; Innflutt:
;;;
;;; Notkun: p=c.i
;;; Fyrir: c er ræð tala
;;; Eftir: p er teljari c
;;;
;;; Notkun: q=c.j
;;; Fyrir: c er ræð tala
;;; Eftir: q er nefnarii c
;;;
;;; Notkun: z=a.+b
;;; Fyrir: a og b eru ræðar tölur
;;; Eftir: z er ný ræð tala, z = a+b
;;;
;;; Notkun: z=a.-b
;;; Fyrir: a og b eru ræðar tölur
;;; Eftir: z er ný ræð tala, z = a-b
;;;
;;; Notkun: z=a.*b
;;; Fyrir: a og b eru ræðar tölur
;;; Eftir: z er ný ræð tala, z = a*b
;;;
;;; Notkun: z=a./b
;;; Fyrir: a og b eru ræðar tölur
;;; Eftir: z er ný ræð tala, z = a/b
;;;
;;; Notkun: x=a+b
;;; Fyrir: a og b eru heiltölur
;;; Eftir: x er ný heiltala, x = a+b
;;;
;;; Notkun: x=a-b
;;; Fyrir: a og b eru heiltölur
```

```
;;; Eftir: x er ný rauntala, x = a-b
;;;
;;; Notkun: x=a+b
;;; Fyrir: a og b eru heiltölur
;;; Eftir: x er ný heiltala, x = a*b
;;;
;;; Notkun: x=a+b
;;; Fyrir: a og b eru heiltölur
;;; Eftir: x er ný heiltala, x = a/b
```

Hrefna svaraði.
17. Hvert þarf sambandið milli forskilyrða og eftirskilyrða í undirklösum og yfirklösum að vera til að röksemdafærsla sé traust?

Forskilyrði fyrir boð í undirklasa (F_B) er afleiðing af forskilyrði fyrir samsvarandi boð í yfirklasa (F_A) þ.e. $F_A \rightarrow F_B$. Forskilyrði fyrir boð í undirklasa má vera veikara (víðara) og gera minni kröfur en samsvarandi boð í yfirklasa, en það má aldrei vera sterkara (þrengra) og gera meiri kröfur.

Anna svaraði
18. Skrifið einingu í Morpho fyrir hlaða. Bæði skal skrifa hönnun og síð. Gleymið ekki fastayrðingu gagna. Þið ráðið hvort einingin er hlutbundin.
[Sjá á vikublaði fv9]

Einfaldur hlutbundinn hlaði
```
; ; ; Hönnun
; ; ;
; ; ; Útflutt
; ; ;
; ; ; Notkun : s = stack( ) ;
; ; ; Fyrir: Ekkert .
; ; ; Eftir: s er nýr tómur hlaði með pláss
; ; ; fyrir ótakmarkaðan fjölda gilda
; ; ; meðan minnis rými tölvunnar leyfir.
; ; ;
; ; ; Innflutt
; ; ;
; ; ; Notkun : s . push ( x ) ;
; ; ; Fyrir: s er hlaði .
; ; ; Eftir: Búið er að setja x ofan á x .
; ; ;
; ; ; Notkun : x = s . pop ( ) ;
; ; ; Fyrir: s er hlaði , ekki tómur .
; ; ; Eftir: x er gildið sem var efst á s,
; ; ; það hefur verið fjarlægt af s.
; ; ;
; ; ; Notkun : b = s.isEmpty();
; ; ; Fyrir: s er hlaði .
; ; ; Eftir: b er true ef s er tómur, annars false

" s t a c k .mmod" =
{{
        ;;;sjá vikublað 9
```

}}
;

Hluti IV: Blandað efni

Ekki þarf endilega að svara neinu dæmi úr þessum hluta, en ekki gleyma að svara 12 dæmum í heild.

Hrefna svaraði

19. Gerið grein fyrir þremur af eftirfarandi aðferðum til viðfangaflutninga:

- Gildisviðföngum

Gildisviðfang (call-by-value) er gildað (evaluated) áður en kallað er á viðkomandi stef, gildið sem út kemur er sett á viðeigandi stað inn í nýju vakningarfærsluna (*activation record*) sem verður til við kallið. Flest forritunarmál styðja gildisviðföng.

- Tilvísunarviðföngum

Tilvísunarviðfang (*call-by-reference*) verður að vera breyta eða ígildi breytu (t.d. stak í fylki). Það er ekki gildað áður en kallað er heldur er vistfang breytunnar sett á viðeigandi stað í nýju vakningafærsluna. Þegar viðfangið er notað inni í stefinu sem kallað er á er gengið beint í viðkomandi minnissvæði, gegnum vistfangið sem sent var.

- Afritsviðföngum

Afritsviðfang (*call-by value/result*, einnig kallað *copy-in/copy-out)* verður að vera breyta, eins og tilvísunarviðfang. Afritsviðfang er meðhöndlað eins og gildisviðföng, nema að þegar kalli lýkur er afritað til baka úr vakningarfærslunni aftur í breytuna.

- Nafnviðföngum eða lötum viðföngum

Nafnviðföng (*call-by-name*): Þegar kallað er á fall eða stef er ekki reiknað úr nafnaviðföngum áður en byrjað er að reikna inn í fallinu eða stefinu sem kallað er á, heldur er reiknað úr hverju viðfangi í hvert skipti sem það er notað.
Löt viðföng *(call-by-need)*: Eins og nafnviðföng, nema hvað að aðeins er reiknað einu sinni, í fyrsta skiptið sem viðfangið er noað.

Dagmar svarar

20. Nefnið þrjár ruslasöfnunaraðferðir og lýsið einni þeirra. Lýsið einnig kostum hennar og göllum, sérstaklega með tilliti til minnisnotkunar og tímaflækju.
Svar: Tilvísunartalning (e. reference counting), merkja og sópa (e. mark and sweep), afritunaraðferð (e. stop and copy).
Tilvísunartalning felst í því að í hverju minnissvæði í kös er teljari, sem ávallt inniheldur fjölda tilvísana á þann hlut úr breytum í forritinu eða öðrum hlutum. Þegar þessi teljari verður núll þá má skila minnissvæðinu.

Kamilla svaraði

21. Svarið báðum eftirfarandi:

- Gefin er eftirfarandi EBNF míllýsing:

```
Exp = 't'
        | '(', Exp, ')'
        | Exp, '+', Exp
        | Exp, '*'
        | '-', Exp
        ;
```

Hverjir eftirfarandi strengja eru í málinu (tvö röng svör gefa núll):
a) (t+2) - er ekki í málinu því 2 er ekki í málinu
b) t** - er í málinu
c) t+ - er ekki í málinu

d) t+t* - er í málinu
e) (t*)+t - er í málinu
f) -t** - er í málinu
g) t-t* - er ekki í málinu, því að eina tvíundaraðgerðin má vera ‘+’
h) (-t)-t - er ekki í lagi því eina tvíundaraðgerðin má vera ‘+’

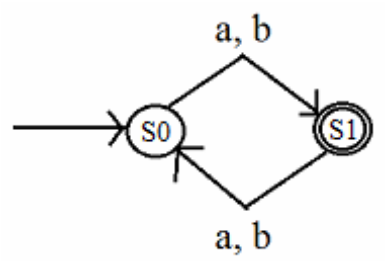
- Sýnið BNF, EBNF eða málrit fyrir mál löglegra formúlna með svigum, tölunni 1 og tvíundaraðgerðinni +. Dæmi um löglegar formúlur eru þá t.d. 1, (1) og 1 + 1, en ekki t.d. 11 + 1 (því 11 er þá ekki lögleg tala).
  BNF: <E>::= 1 | (<E>) | <E>+<E>
  EBNF: E = ‘1’ | ‘(‘, E, ‘)’ | E, ‘+’, E;

Anna svaraði

22. Svarið báðum eftirfarandi:

- Teiknið endanlega stöðuvél (löggenga eða briðgenga) fyrir málið yfir stafrófið *{a,b}* sem innihalda strenginn *abbababb* sem hlutstreng.

  stöðuvel með 2 stöðum a,b í s1 og a,b í s0, s1 er lokastaða



- Skrifið reglulega segð fyrir mál þeirra strengja yfir stafrófið {a,b} sem innihalda strenginn abbababb sem hlutstreng.

  (a|b)*abbababb(a|b)*