TÖL304G Forritunarmál
Lokapróf haustið 2012

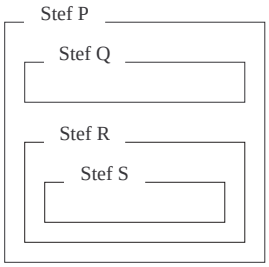
Svarið 10 af eftirfarandi dæmum og a.m.k. tveimur úr hverjum af köflum I–III. Ef þið svarið fleiri en 10 dæmum skuluð þið tilgreina hvaða svör eiga að gilda til einkunnar. Ef þið tilgreinið ekki slík svör og svarið fleiri en 10 verður tekið meðaltal allra þeirra dæma sem þið reynið við. Öll dæmi gilda jafnt. **Skrifið á fremstu hvítu blaðsíðu úrlausnarbókar hvaða lausnir eiga að gilda til einkunnar.**

Engin hjálpargögn eru leyfileg.

Hluti I: Bálkmótun o.fl.

1. Lýsið því hvernig vakningarfærslur í bálkmótuðum forritunarmálum eru tengdar saman með stýrihlekkjum (*control link, dynamic link*) og tengihlekkjum (*aðgangshlekkjum, access link, static 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 sýnir. Hverjar eftirfarandi fullyrðinga eru þá sannar og hverjar eru ósannar? Eitt rangt svar veldur því að ekkert fæst fyrir dæmið.



(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 R.

(e) Kalla má á P úr S.

(f) Kalla má á P úr Q.

(g) Kalla má á Q úr R.

(h) Kalla má á Q úr S.

(i) Kalla má á Q úr P.

(j) Kalla má á R úr Q.

(k) Kalla má á R úr S.

(l) Kalla má á R úr P.

(m) Í P má nota staðværar breytur úr Q.

(n) Í P má nota staðværar breytur úr R.

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

(p) Í Q má nota staðværar breytur úr P.

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

(r) Í Q má nota staðværar breytur úr S.

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

(t) Í S má nota staðværar breytur úr R.

(u) Í S má nota staðværar breytur úr Q.

(v) Í R má nota staðværar breytur úr P.

(w) Í R má nota staðværar breytur úr S.

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

3. Rökstyðjið að til þess að bálkmótað forritunarmál bjóði þann möguleika að falli sé skilað úr falli þurfi vakningarfærslur að vera geymdar í kös (heap) frekar en á hlaða (stack). Lýsið þeim vandamálum sem upp koma ef vakningarfærslur eru á hlaða.

4. Svarið öllum eftirfarandi:

   (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 verða að vera geymdar í kös.

   (d) Rökstyðjið að í síðarnefndu tveimur forritunarmálunum sé nauðsynlegt að vakningarfærslur séu í kös.

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

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

int i,a[10];
for( i=0 ; i!=10 ; i++ ) a[i]=i;
f(a[0],a[a[0]]);
print a[0], a[1], a[2];
```

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

   - gildisviðföng?
   - tilvísunarviðföng?
   - nafnviðföng?

Hluti II: Listavinnsla o.fl.

6. Skrifið fall í Scheme, CAML, Morpho eða Haskell sem tekur eitt viðfang sem er listi lista af fleytitölum milli 0 og 1 og skilar tölu sem er minnsta hágildi innri listanna, þ.e. minnst af þeim tölum sem fást þegar fundin er hæsta tala í hverjum innri lista. Þið megið reikna með því að hæsta gildi í tómamenginu sé 0 og lægsta gildi í tómamenginu sé 1.

7. Skrifið halaendurkvæmt fall í Scheme, Haskell, CAML eða Morpho, sem tekur eina heiltölu $n \geq 0$ sem viðfang og skilar listanum $(1\ 2\ \ldots n)$. Ef fallið er ekki halaendurkvæmt fæst aðeins hálft fyrir dæmið.

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

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

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

10. Skrifið tvö föll í Scheme, CAML, Morpho eða Haskell sem eru hefðbundnar útfærslur á insert-left og insert-right, þ.e. föll sem taka þrjú viðföng og beita tvíundaraðgerð frá vinstri til hægri eða frá hægri til vinstri.

Hluti III: Einingaforritun o.fl.

11. Gerið ráð fyrir að til séu klasar A og B í einhverju hlutbundnu forritunarmáli og að B sé undirklasi A og að báðir klasarnir innihaldi tilviksboð t með eftirfarandi lýsingar í A og B.

    - Lýsing í A:

      Notkun: y = z.t(x);
      Fyrir: $a \leq x \leq b$.
      Eftir: $c \leq y \leq d$.
    - Lýsing í B:

      Notkun: y = z.t(x);
      Fyrir: $e < x < f$.
      Eftir: $g < y < h$.

    Hvað þarf sambandið að vera milli a, b, c, d, e, f, g og h til að rökfræðilegt samband klasanna A og B sé rétt? Sambandið sem þið lýsið skal vera nauðsynlegt og nægjanlegt.

12. Skrifið hönnunarskjal fyrir einingu í Morpho, Java eða C++ sem útfærir biðröð (*queue*) gilda (eða hluta). Einingin skal vera nægilega fjölnota til að unnt sé að búa til biðröð hluta af hvaða tagi sem er sem uppfyllir eðlileg skilyrði, en þó skal nýta þá tögun sem forritunarmálið býður upp á til að unnt sé að tryggja að biðröðin innihaldi aðeins gildi af því tagi sem til er ætlast.

    Munið upplýsingahuld. Aðeins þarf hönnunarskjal, enga útfærslu þarf.

13. Skrifið hönnunarskjal fyrir fjölnota einingu fyrir forgangsbiðraðir í Java, C++ eða Morpho.

    Munið upplýsingahuld.

    Skrifið einnig fastayrðingu gagna og útfærið aðgerðina til að fjarlægja gildi úr forgangsbiðröðinni.

14. Skrifið hönnunarskjal fyrir fjölnota einingu í Morpho sem útfærir tvinntölureikninga. Úr einingunni skulu vera útfluttar aðgerðir til að reikna með tvinntölur og innfluttar skulu vera aðgerðir til reikninga með rauntölur.

Hluti IV: Blandað efni

15. Lýsið ruslasöfununaraðferðinni sem byggist á tilvísunartalningu. Nefnið einhverja ástæðu þess að tilvísunartalning er oftast hægvirkari en aðrar aðferðir. Nefnið ástæðu þess að þrátt fyrir lélegan hraða er tilvísunartalning oft notuð í forritunarmálum s.s. C++.

16. Gefin er eftirfarandi BNF mállýsing:

```
<E> ::= <T> <Em>
<Em> ::= + <T> <Em>
<Em> ::= ϵ
<T> ::= <F> <T>
<T> ::= ϵ
<F> ::= <F> !
<F> ::= ( <E> )
<F> ::= t
<F> ::= f
```

    Hverjir eftirfarandi strengja eru í málinu (eitt rangt svar gefur núll)?

    (a) `f+t`
    (b) `!f+t`
    (c) `f!+t`
    (d) `f+!t`
    (e) `f+t!`
    (f) `ft`
    (g) `!ft*t`
    (h) `ft!*t`
    (i) `f!t`
    (j) `ft!`
    (k) `(ft)`
    (l) `(f+t)!`
    (m) `(f+tf)!`

17. Teiknið endanlega stöðuvél (löggenga eða brigðgenga) fyrir málið yfir stafrófið $\{(,)\}$ sem inniheldur þá strengi sem hafa sviga í jafnvægi og dýpt sviga er í mesta lagi 3. Strengirnir '(())' og '()()((()()))' eiga að vera í málinu en ekki '((()))'.

18. Hvað er sniðmengi málanna sem hafa eftirfarandi BNF mállýsingar?

    Mál A:

    ```
    <C> ::= <B> <A>
    <B> ::= c <B> b
    <B> ::= ϵ
    <A> ::= a <C>
    <A> ::= ϵ
    ```

    Mál B:

    ```
    <C> ::= <B> <A>
    <B> ::= c <B>
    <B> ::= ϵ
    <A> ::= b <C> a
    <A> ::= ϵ
    ```

19. Sýnið BNF, EBNF eða málrit fyrir mál λ-reiknisegða með svigum, breytunöfnunum a, b og c, fallsbeitingu NM þar sem N og M eru λ-reiknisegðir, og fallsskilgreiningum $\lambda x.N$ þar sem x er eitt breytunafnanna og N er λ-reiknisegð. Dæmi um löglegar formúlur eru þá t.d. $\lambda a.ab$, $a(ab)$, abc og $(\lambda a.ba)c$, en ekki t.d. x eða $\lambda ab.c$.

20. Íhugið málin sem skilgreind eru annars vegar með reglulegu segðinni $(x(x(x(xy)^*y)^*y)^*y)^*$ og hins vegar með BNF mállýsingunni (athugið að ϵ stendur fyrir tóma strenginn) að neðan.

    S ::= x <S> y <S> | ϵ

    (a) Segið til um hvort reglulega segðin og BNF mállýsingin lýsa sama máli.

    (b) Ef málin eru ekki þau sömu, lýsið þá báðum málunum í stuttum texta og sýnið streng sem er í öðru málinu en ekki hinu.

    (c) Ef málin eru þau sömu lýsið þá málinu sem þau lýsa í stuttum texta.

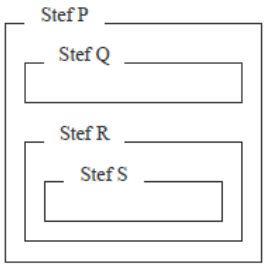
TÖL304G Forritunarmál
Lokapróf haustið 2012

Svarið 10 af eftirfarandi dæmum og a.m.k. tveimur úr hverjum af köflum I–III. Ef þið svarið fleiri en 10 dæmum skuluð þið tilgreina hvaða svör eiga að gilda til einkunnar. Ef þið tilgreinið ekki slík svör og svarið fleiri en 10 verður tekið meðaltal allra þeirra dæma sem þið reynið við. Öll dæmi gilda jafnt. Skrifið á fremstu hvítu blaðsíðu úrlausnarbókar hvaða lausnir eiga að gilda til einkunnar.
Engin hjálpargögn eru leyfileg.

Hluti I: Bálkmótun o.fl.

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

2. Gerið ráð fyrir földun í bálkmótuðu forritunarmáli eins og myndin sýnir. Hverjar eftirfarandi fullyrðinga eru þá sannar og hverjar eru ósannar? Eitt rangt svar veldur því að ekkert fæst fyrir dæmið.



Lausn:
(a) Kalla má á S úr P. **- Ósatt**
(b) Kalla má á S úr Q. **- Ósatt**
(c) Kalla má á S úr R. **- Satt**
(d) Kalla má á P úr R. **- Satt**
(e) Kalla má á P úr S. **- Satt**
(f) Kalla má á P úr Q. **- Satt**
(g) Kalla má á Q úr R. **- Satt**
(h) Kalla má á Q úr S. **- Satt**
(i) Kalla má á Q úr P. **- Satt**
(j) Kalla má á R úr Q. **- Satt**
(k) Kalla má á R úr S. **- Satt**
(l) Kalla má á R úr P. **- Satt**
(m) Í P má nota staðværar breytur úr Q. **- Ósatt**

(n) Í P má nota staðværar breytur úr R. **- Ósatt**
(o) Í P má nota staðværar breytur úr S. **- Ósatt**
(p) Í Q má nota staðværar breytur úr P. **- Satt**
(q) Í Q má nota staðværar breytur úr R. **- Ósatt**
(r) Í Q má nota staðværar breytur úr S. **- Ósatt**
(s) Í S má nota staðværar breytur úr P. **- Satt**
(t) Í S má nota staðværar breytur úr R. **- Satt**
(u) Í S má nota staðværar breytur úr Q. **- Ósatt**
(v) Í R má nota staðværar breytur úr P. **- Satt**
(w) Í R má nota staðværar breytur úr S. **- Ósatt**
(x) Í R má nota staðværar breytur úr Q. **- Ósatt**

3. Rökstyðjið að til þess að bálkmótað forritunarmál bjóði þann möguleika að falli sé skilað úr falli þurfi vakningarfærslur að vera geymdar í kös (heap) frekar en á hlaða (stack). Lýsið þeim vandamálum sem upp koma ef vakningarfærslur eru á hlaða.
Lausn:
Þegar falli er skilað úr falli í bálkmótuðu forritunarmáli þá er í raun verið að skila lokun sem inniheldur upplýsingar sem nægja til að kalla á fallið í réttu samhengi.
Lokun inniheldur fallsbendi og aðgangshlekk. Fallsbendirinn vísar á fallið sem átti að skila. Aðgangshlekkurinn vísar á vakningarfærslu þess stefs sem innihélt viðkomandi fall textalega séð, þ.e. vakningarfærsluna sem inniheldur breyturnar í næstu földunarhæð fyrir ofan.
Þegar lokun er framkvæmd þá er búin til ný vakningarfærsla fyrir kall á fallið í lokuninni. Aðgangshlekkurinn í lokuninni er notaður sem aðgangshlekkurinn í vakningarfærslunni. Þá er fallið framkvæmt í réttu umhverfi og kemst í breyturnar í ytri föllum (næstu földunarhæðum fyrir ofan) á þeim stað og í því samhengi þar sem fallið var upphaflega skilgreint.
Vélamálsþulan á bakvið föll er ávallt til staðar á meðan forrit er í keyrslu. Fallbendirinn í lokun vísar á þulu falls. Það er ekkert áhyggjuefni að þulan hætti að vera til. Hinsvegar er ekki hægt að segja það sama um vakningarfærslur.
Aðgangshlekkur í lokun vísar í vakningarfærslu. Lokun er aðeins í nothæfu ástandi á meðan sú vakningarfærsla er enn til staðar. Þegar vakningarfærslur eru geymdar á hlaða þá er minnissvæði vakningarfærslu skilað og vakningarfærslan hættir að vera til þegar það er snúið til baka úr vakningu fallsins. Þegar næsta kall er framkvæmt þá er minnisvæðið endurnýtt þegar ný vakningarfærsla er sett efst á hlaðann. Það er því ekki hægt að skila falli úr falli ef vakningarfærslur eru geymdar á hlaða. Ef vakningarfærslur eru geymdar í kös (heap) frekar en á hlaða (stack) þá eru þær geymdar í víðværu minni og geta haldið áfram að vera til staðar þó þær séu ekki hluti af stýrikeðjunni.

4. Svarið öllum eftirfarandi:
(a) Hvaða upplýsingar eru geymdar í vakningarfærslum forritunarmála?
Lausn:
Staðværar breyrur
Stýrihlekkir
Tengihlekkir
Viðföng

(b) Hverjar þessara upplýsinga eru aðeins í bálkmótuðum forritunarmálum og hvers vegna?
Lausn:
Tengihlekkir eru bara í bálkmótuðum forritunarmálum. Ástæðan er sú að tengihlekkur er notaður til aðgangs að breytum í efri földunarhæðum og er því merkingarlaus ef ekki er um bálkmótun að ræða.

(c) Nefnið tvö forritunarmál þar sem vakningarfærslur eru yfirleitt geymdar á hlaðanum og tvö þar sem vakningarfærslur verða að vera geymdar í kös.
Lausn:
Vakningarfærslur á hlaða: C og Java. Vakningarfærslur í kös: Scheme og Morpho.
Ein veigamikil afleiðing þess að geyma vakningarfærslur í kös er að þá er auðveldlega hægt að útfæra lokanir sem fyrsta-flokks gildi (first-class value) í forritunarmálinu. Þá er hægt að vinna með lokanir eins og hvert annað gildi í forritunarmálinu, þar með talið nota það sem viðfang í kalli á fall, skila því úr falli, o.s.frv.
Ef vakningarfærslur eru geymdar á hlaða þá er ekki hægt að vinna með lokanir á jafn sveigjanlegan hátt. Það má til dæmis ekki skila lokun sem niðurstöðu úr kalli (því lokunin bendir á vakningarfærslu og það er ekki hægt að tryggja að sú vakningarfærsla verði áfram til staðar jafn lengi og lokunin er til staðar.)

(d) Rökstyðjið að í síðarnefndu tveimur forritunarmálunum sé nauðsynlegt að vakningarfærslur séu í
Lausn:
Scheme og Morpho eru bálkmótuð og leyfa okkur að skila fallsgildum úr föllum. Slík fallsgildi eru lokanir sem innihalda aðgangshlekk sem vísar á vakningarfærslu. Lokun er bara nothæf svo lengi sem sú vakningarfærsla er enn til staðar. Það er nauðsynlegt að geyma vakningarfærslur í kös til að tryggja að þær geti lifað jafn lengi og lokanir sem vísa á þær.

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

```
void f(x,y)
{
        x = 2;
        print x,y;
        y = 1;
}
int i, a[10];
for( i = 0 ; i != 10 ; i++ ) a[i] = i;
f( a[0], a[ a[0] ]);
print a[0], a[1], a[2];
```

Hvað skrifar þetta forrit (fimm gildi í hvert skipti) ef viðföngin eru
Gildisviðföng
Lausn:
2, 0, 0, 1, 2

Tilvísunarviðföng
Lausn:
2, 2, 1, 1, 2

Nafnviðföng
Lausn:
2, 2, 2, 1, 1

Hluti II: Listavinnsla o.fl.

6. Skrifið fall í Scheme, CAML, Morpho eða Haskell sem tekur eitt viðfang sem er listi lista af fleytitölum milli 0 og 1 og skilar tölu sem er minnsta hágildi innri listanna, þ.e. minnst af þeim tölum sem fást þegar fundin er hæsta tala í hverjum innri lista. Þið megið reikna með því að hæsta gildi í tómamenginu sé 0 og lægsta gildi í tómamenginu sé 1.

Lausn:

Lausn í Haskell:

```
-- Notkun: minmax xs
-- Fyrir: xs er listi [x1,x2,x3,..,xn] þar sem x_i er listi af fleytitölum milli 0 og 1.
-- Gildi: Tala sem er minnsta hágildi innri listanna, þ.e. min(max(x1),max(x2),max(x3),..,max(xn))
minmax xs = minimum $ map maximum xs
```

Lausn í Haskell sem ræður við tómalistann:
Gerum ráð fyrir að hæsta gildi í tómamenginu sé 0 og lægsta gildi í tómamenginu sé 1, sbr. texta í dæminu sjálfu.

```
-- Notkun: minmax xs
-- Fyrir: xs er listi [x1,x2,x3,..,xn] þar sem x_i er listi af fleytitölum milli 0 og 1.
-- Gildi: Tala sem er minnsta hágildi innri listanna, þ.e. min(max(x1),max(x2),max(x3),..,max(xn))
minmax xs = xmin $ map xmax xs
        where
                xmin [] = 1.0
                xmin xs = minimum xs
                xmax [] = 0.0
                xmax xs = maximum xs
```

Lausn í CAML sem ræður við tómalistann:
Gerum ráð fyrir að hæsta gildi í tómamenginu sé 0 og lægsta gildi í tómamenginu sé 1, sbr. texta í dæminu sjálfu.

```
(* Notkun: minmax xs
* Fyrir: xs er listi [x1,x2,x3,..,xn] þar sem x_i er listi af fleytitölum milli 0 og 1.
* Gildi: Tala sem er minnsta hágildi innri listanna, þ.e. min(max(x1),max(x2),max(x3),..,max(xn)) *)
let minmax xs =
        let maximum x = List.fold_left max 0.0 x in
                let maxlist = List.map maximum xs in
                        List.fold_left min 1.0 maxlist;;
```

7. Skrifið halaendurkvæmt fall í Scheme, Haskell, CAML eða Morpho, sem tekur eina heiltölu n => 0 sem viðfang og skilar listanum (1 2 . . . n). Ef fallið er ekki halaendurkvæmt fæst aðeins hálft fyrir dæmið.

Lausn:

Lausn í Haskell:

```
-- Notkun: index n
-- Fyrir: n er heiltala, n >= 0
-- Gildi: Listinn [1,2,..,n]
index n = [1..n]
```

Lausn í Haskell sem býr til listann halaendurkvæmt:

```
-- Notkun: index n
-- Fyrir: n er heiltala, n >= 0
```

```
-- Gildi: Listinn [1,2,..,n]
index n = h n []
        where
                h 0 u = u
                h n u = h (n-1) (n:u)
```

Lausn í Morpho:

```
-- Notkun: index n
-- Fyrir: n er heiltala, n >= 0
-- Gildi: Listinn [1,2,...,n]
index = fun(n)
{
        var x = [];
        var i = n;
        while (i > 0)
        {
                // x = [i-1,...,n]
                x = i:x;
                i = i-1;
        };
        return x;
};
```

Lausn í Scheme:

```
;; Notkun: (index n)
;; Fyrir: n er heiltala, n >= 0
;; Gildi: Listinn (1 2 ... n)
(define (index n)
        (define (hjalp i acc)
                (if (= i 0)
                        acc
                        (hjalp (- i 1) (cons i acc))
                )
        )
        (hjalp n '())
)
```

8. Skrifið halaendurkvæmt1 fall í Scheme, CAML eða Morpho, sem tekur eina heiltölu n => 0 sem viðfang og skilar listanum [n, n - 1, ..., 1].

Lausn:

Lausn í Haskell:

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

Lausn í Morpho:

```
-- Notkun: rindex n
-- Fyrir: n er heiltala, n >= 0
-- Gildi: Listinn [n,n-1,..,1]
rindex = fun(n)
```

```
{
	var x = [];
	var i = 0;
	while (i <= n)
	{
		// x = [i-1,..,1]
		x = i:x;
		i = i+1;
	};
	return x;
};
```

9. Forritið straum í Scheme eða Morpho sem inniheldur óendanlegu rununa [1, 3, 5, 7; ...] af öllum jákvæðum oddatölum. Ef ykkur vantar hjálparföll verðið þið að skilgreina þau.
Lausn:
Lausn í Scheme

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

10. Skrifið tvö föll í Scheme, CAML, Morpho eða Haskell sem eru hefðbundnar útfærslur á insert-left og insert-right, þ.e. föll sem taka þrjú viðföng og beita tvíundaraðgerð frá vinstri til hægri eða frá hægri til vinstri.
Lausn:
Lausn í Haskell:

```
-- Notkun: foldl f u xs
-- Fyrir: f er tvíundarfall, u er löglegt vinstra viðfang í f, x=[x1,..,xN] er listi gilda sem eru lögleg, hægri viðföng í f
-- Gildi: Margfeldi u og gildanna í x, reiknað frá, vinstri til hægri, þ.e. f( ... f(f(u,x1),x2) ... xN)
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl _ u [] = u
foldl f u (x:xs) = foldl f (f u x) xs

-- Notkun: foldr f u xs
-- Fyrir: f er tvíundarfall, u er löglegt hægra viðfang í f, x=[x1,..,xn] er listi gilda sem eru lögleg, vinstri viðföng í f
-- Gildi: Margfeldi u og gildanna í x, reiknað frá, hægri til vinstri, þ.e. f(x1,f(x2,f(x3, ... f(xN,u))))
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr _ u [] = u
foldr f u (x:xs) = f x $ foldr f u xs
```

Hluti III: Einingaforritun o.fl.

11. Gerið ráð fyrir að til séu klasar A og B í einhverju hlutbundnu forritunarmáli og að B sé undirklasi A og að báðir klasarnir innihaldi tilviksboð t með eftirfarandi lýsingar í A og B.

Lýsing í A:
Notkun: y = z.t(x);
Fyrir: a <= x <= b.
Eftir: c <= y <= d.
Lýsing í B:
Notkun: y = z.t(x);
Fyrir: e < x < f.
Eftir: g < y < h.

Hvað þarf sambandið að vera milli a; b; c; d; e; f; g og h til að rökfræðilegt samband klasanna A og B sé rétt? Sambandið sem þið lýsið skal vera nauðsynlegt og nægjanlegt.
Lausn:
Látum FA og FB standa fyrir forskilyrðin í A og B, þ.e. F_A = a <= x <= b og F_B = e < x < f.
Önnur leið til að hugsa FA og FB er að FA sé y 2 [a, b] og FB sé y ε (e, f).
Það verður að gilda að FA → FB, þ.e. ef fullyrðingin í forskilyrði A er sönn þá er fullyrðingin í forskilyrði B einnig sönn. Forskilyrðið FB í undirklasanum má því vera víðara en forskilyrðið FA í yfirklasanum.

FA → FB
x 2 [a, b] → x ε (e, f)

Til að þetta gildi alltaf þá verður eftirfarandi samband að gilda:

e < a <= x <= b < f

Látum EA og EB standa fyrir eftirskilyrðin í A og B, þ.e. EA = c <= y <= d og EB = g < y < h.
Önnur leið til að hugsa þetta er að EA sé y 2 [c, d] og að EB sé y 2 (g, h).
Það verður að gilda að EB =) EA, þ.e. ef fullyrðingin í eftirskilyrði B er sönn þá er fullyrðingin í eftirskilyrði A einnig sönn. Eftirskilyrðið EB í undirklasanum má vera þrengra en eftirskilyrðið EA í grunnklasanum.

EB → EA
y 2 (g, h) → y ε [c, d]

Til að þetta gildi alltaf þá verður eftirfarandi samband að gilda:

c <= g < y < h <= d

12. Skrifið hönnunarskjal fyrir einingu í Morpho, Java eða C++ sem útfærir biðröð (queue) gilda (eða hluta). Einingin skal vera nægilega fjölnota til að unnt sé að búa til biðröð hluta af hvaða tagi sem er sem uppfyllir eðlileg skilyrði, en þó skal nýta þá tögun sem forritunarmálið býður upp á til að unnt sé að tryggja að biðröðin innihaldi aðeins gildi af því tagi sem til er ætlast.
Munið upplýsingahuld. Aðeins þarf hönnunarskjal, enga útfærslu þarf.
Lausn:
Lausn í Morpho:

```
;;; Hönnun
;;;
;;; Útflutt:
;;;
;;; Notkun: q = queue()
;;; Fyrir: Ekkert
;;; Eftir: q er ný tóm biðröð gilda af tagi T
;;;
;;; Innflutt:
;;;
;;; Notkun: q.insert(x);
```

```
;;; Fyrir: q er biðröð, x er gildi af tagi T
;;; Eftir: Búið er að setja x aftast í biðröðina q
;;;
;;; Notkun: x = q.remove();
;;; Fyrir: q er biðröð, q má ekki vera tóm
;;; Eftir: x er gildið sem var fremst í q,
;;; búið er taka x úr q
;;;
;;; Notkun: x = q.empty();
;;; Fyrir: q er biðröð
;;; Eftir: x er satt þþaa. q er tóm
;;;
```

13. Skrifið hönnunarskjal fyrir fjölnota einingu fyrir forgangsbiðraðir í Java, C++ eða Morpho.
Munið upplýsingahuld.
Skrifið einnig fastayrðingu gagna og útfærið aðgerðina til að fjarlægja gildi úr forgangsbiðröðinni.
Lausn:
Hönnun í Morpho:

```
;;; Hönnun
;;;
;;; Útflutt:
;;;
;;; Notkun: q = pq()
;;; Fyrir: Ekkert
;;; Eftir: q er ný tóm forgangsbiðröð gilda af tagi T,
;;; þar sem forgangur er skilgreindur með <<==.
;;;
;;; Innflutt:
;;;
;;; Notkun: z = x <<== y
;;; Fyrir: x og y eru gildi af tagi T
;;; 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.
;;; Þ.e. <<== ætti að vera hlutröðunarvensl.
;;;
;;; Notkun: q.insert(x);
;;; Fyrir: q er forgangsbiðröð, x er gildi sem má setja í q
;;; Eftir: Búið er að setja x í q
;;;
;;; Notkun: x = q.remove();
;;; Fyrir: q er forgangsbiðröð, q má ekki vera tóm
;;; Eftir: x er gildið sem var fremst í q,
;;; búið er taka x úr q
;;;
;;; Notkun: x = q.empty();
;;; Fyrir: q er forgangsbiðröð
;;; Eftir: x er satt þþaa. q er tóm
;;;
```

Fastayrðing og insert í Morpho:

```
pq = obj()
{
	;;; Fastayrðing gagna:
	;;; Stökin í forgangsbiðröðinni eru geymd í listanum q
	;;; í engri sérstakri röð.
	var q = [];
	msg insert(x)
	{
		q = x:q;
	}
};
```

14. Skrifið hönnunarskjal fyrir fjölnota einingu í Morpho sem útfærir tvinntölureikninga. Úr einingunni skulu vera útfluttar aðgerðir til að reikna með tvinntölur og innfluttar skulu vera aðgerðir til reikninga með rauntölur.

Lausn:

Lausn í Morpho:

```
;;; 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
```

Hluti IV: Blandað efni

15. Lýsið ruslasöfununaraðferðinni sem byggist á tilvísunartalningu. Nefnið einhverja ástæðu þess að tilvísunartalning er oftast hægvirkari en aðrar aðferðir. Nefnið ástæðu þess að þrátt fyrir lélegan hraða er tilvísunartalning oft notuð í forritunarmálum s.s. C++.

Lausn:

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.

Í hvert skipti sem ný tilvísun er búin til eða tilvísun er eytt, þá þarf að uppfæra viðeigandi teljara. Ef það er mikið um tilvísanir sem lifa í skamman tíma þá er mikið um uppfærslur á teljurum. Þetta gerir tilvísunartalningu hægvirkari en margar aðrar aðferðir.

Þær tilvísanir sem lifa vanalega skemmst eru tilvísanir frá staðværum breytum. Einföld for-lykkja sem gengur yfir öll stök í tengdum lista með staðværri breytu sem vísar á núverandi stak í listanum, veldur því að teljari í hverjum hlekk er hækkaður þegar sá hlekkur er heimsóttur, og síðan lækkaður aftur þegar við færum okkur yfir á næsta hlekk.

Þrátt fyrir lélegan hraða er tilvísunartalning oft notuð því hún er einföld í útfærslu og það er oftast hægt að útfæra hana innan forritunarmála, án þess að þurfa að breyta sjálfu forritunarmálinu.

Sem dæmi má nefna C++, það er engin innbyggð ruslasöfnun í C++, en það eru til ýmis forritunarsöfn (libraries) fyrir C++ sem útfæra ruslasöfnun. Það er oftast gert með tilvísunartalningu og svokölluðum snjöllum bendum (smart pointers) sem sjá um að viðhalda tilvísunartalningu.

16. Gefin er eftirfarandi BNF mállýsing:

```
<E> ::= <T> <Em>
<Em> ::= + <T> <Em>
<Em> ::= ε
<T> ::= <F> <T>
<T> ::= ε
<F> ::= <F> !
<F> ::= ( <E> )
<F> ::= t
```

<F> ::= f

Hverjir eftirfarandi strengja eru í málinu (eitt rangt svar gefur núll)?

(a) f+t
(b) !f+t
(c) f!+t
(d) f+!t
(e) f+t!
(f) ft
(g) !ft*t
(h) ft!*t
(i) f!t
(j) ft!
(k) (ft)
(l) (f+t)!
(m) (f+tf)!

Lausn:

(a) f+t
Svar: Já
Útleiðsla:

E → T Em
→ T + T Em
→ T + T
→ F + F
→ f + t

(b) !f+t
Svar: Nei, til að fá ! þarf að nota regluna F → F!, en það er engin epsilon regla fyrir F.

(c) f!+t
Svar: Já
Útleiðsla:

E → T Em
→ T + T Em
→ T + T
→ F + F
→ F! + F
→ f! + t

(d) f+!t
Svar: Nei, til að fá ! þarf að nota regluna F *!* F!, en það er engin epsilon regla fyrir F. Endar með f+<F>!t og getur ekki losnað við F.

(e) f+t!
Svar: Já
Útleiðsla:

E → T Em
→ T + T Em
→ T + T
→ F + F
→ F + F!
→ f + t!

(f) f*t
Svar: Já
Útleiðsla:

E → T Em
→ T
→ F Tm
→ F * F Tm
→ F * F
→ f * t

(g) !f*t
Svar: Nei, til að fá ! þarf að nota regluna F → F!, en það er engin epsilon regla fyrir F.

(h) f!*t
Svar: Já
Útleiðsla:

E → T Em
→ T
→ F Tm
→ F * F Tm
→ F * F
→ F! * F
→ f! * t

(i) f*!t
Svar: Nei, til að fá ! þarf að nota regluna F → F!, en það er engin epsilon regla fyrir F. Endar með f*<F>!t og getur ekki losnað við F.

(j) f*t!
Svar: Já
Útleiðsla:

E → T Em
→ T
→ F Tm
→ F * F Tm
→ F * F
→ F * F!
→ f * t!

(k) (f*t)
Svar: Já
Útleiðsla:

E → T Em
→ T
→ F Tm
→ F
→ (E)
→ (T Em)
→ (T)
→ (F Tm)

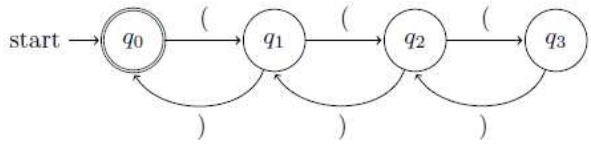
→ (F * F Tm)
→ (F * F)
→ (f * t)

(l) !(f*t)
Svar: Nei, til að fá ! þarf að nota regluna F *!* F!, en það er engin epsilon regla fyrir F.

(m) (f*t)!
Svar: Já
Útleiðsla:

E → T Em
→ T
→ F Tm
→ F
→ F!
→ (E)!
→ (T Em)!
→ (T)!
→ (F Tm)!
→ (F * F Tm)!
→ (F * F)!
→ (f * t)!

17. Teiknið endanlega stöðuvél (löggenga eða brigðgenga) fyrir málið yfir stafrófið { (,) } sem inniheldur þá strengi sem hafa sviga í jafnvægi og dýpt sviga er í mesta lagi 3. Strengirnir ’(( ))’ og ’( ) ( ) (( ) ( )))’ eiga að vera í málinu en ekki ’(((( ))))’.
Lausn:



18. Hvað er sniðmengi málanna sem hafa eftirfarandi BNF míllýsingar?

Mál A:

<C> ::= <B> <A>
<B> ::= c <B> b
<B> ::= ε
<A> ::= a <C>
<A> ::= ε

Mál B:

<C> ::= <B> <A>
<B> ::= c <B>
<B> ::= ε
<A> ::= b <C> a
<A> ::= ε

Lausn:

19. Sýnið BNF, EBNF eða málrit fyrir mál λ-reiknisegða með svigum, breytunöfnunum a, b og c, fallsbeitingu NM þar sem N og M eru λ -reiknisegðir, og fallsskilgreiningum λ x.N þar sem x er eitt

breytunafnanna og N er λ -reiknisegð. Dæmi um löglegar formúlur eru þá t.d. λ a.ab, a(ab), abc og (λ a.ba)c, en ekki t.d. x eða λ ab.c.

Lausn:

BNF:

<E> ::= (<E>) | _<V>.<E> | <V> | <E><E> <V> ::= a | b | c

EBNF:

e = ’(’,e,’)’ | ’_’,v,’.’,e | v | e, e; v ’a’ | ’b’ | ’c’

20. Íhugið málin sem skilgreind eru annars vegar með reglulegu segðinni (x(x(x(xy)*y)*y)*y)* og hins vegar með BNF mállýsingunni (athugið að ε stendur fyrir tóma strenginn) að neðan.

S ::= x <S> y <S> | ε

(a) Segið til um hvort reglulega segðin og BNF mállýsingin lýsa sama máli.

Lausn:

Nei, þetta eru tvö mismunandi mál.

(b) Ef málin eru ekki þau sömu, lýsið þá báðum málunum í stuttum texta og sýnið streng sem er í öðru málinu en ekki hinu.

Lausn:

Reglulega segðin lýsir málinu fyrir stafrófið x,y þar sem x verkar svipað og svigi opnast og y sem svigi lokast og svigar eru í jafnvægi og í mesta lagi fjórir svigar (x) eru opnir á hverjum stað í strengnum. BNF mállýsingin lýsir hinsvegar málinu þar sem svigar eru í jafnvægi og engin takmörk eru á dýpt sviga. Strengurinn xxxxxyyyyy er í BNF málinu en ekki hinu.

(c) Ef málin eru þau sömu lýsið þá málinu sem þau lýsa í stuttum texta.

Lausn:

Á ekki við.