TÖL304G Forritunarmál

Prófdagur og tími: 17.12.2013 09:00-12:00

Prófstaður:
Aðalbygging - A222 (fjöldi: 5)
Aðalbygging - A229 (fjöldi: 1)
Háskólatorg - HT302 (fjöldi: 1)
VR-2 - V138 (fjöldi: 18)
VR-2 - V147 (fjöldi: 16)
VR-2 - V152 (fjöldi: 22)
VR-2 - V155 (fjöldi: 25)
VR-2 - V156 (fjöldi: 23)
VR-2 - V258 (fjöldi: 11)
VR-2 - V261 (fjöldi: 15)

HÁSKÓLI ÍSLANDS

Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild

Skriflegt próf

Skráðir til prófs: 137

Kennari:
Snorri Agnarsson (snorri@hi.is / S: 8613270 / GSM: 8613270) Umsjónarkennari

Kennslumisseri: Haust 2013

Úrlausnir skulu merktar með nafni

Prófbók/svarblöð:
Línustrikuð prófbók

Hjálpargögn:
Engin hjálpargögn eru leyfileg.

Önnur fyrirmæli:

Aðgangur að prófverkefni að loknu prófi:
Kennslusvið sendir eintak í prófasafn

Einkunnir skulu skráðar í Uglu eigi síðar en 07.01.2014.

AHUGIÐ að einhverjar úrlausnir úr fjölmennum prófum geta verið í þunnum umslögum sem auðvelt er að yfirsjást. GÓÐ VINNUREGLA er að byrja á því að opna öll umslög, telja úrlausnir og athuga hvort fjöldi stemmir við uppgefinn fjölda sem kvittað var fyrir.

TÖL304G Forritunarmál
Lokapróf haustið 2013

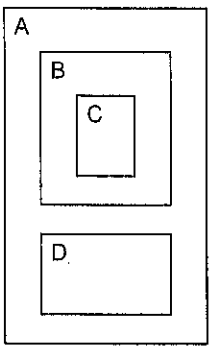
Svarið 10 af eftirfarandi dæmum og a.m.k. tveimur úr hverjum af köflum I–III. Ef þið svarið fleiri en 10 dæmum skuluð þið tilgreina hvaða svör eiga að gilda til einkunnar. Ef þið tilgreinið ekki slík svör og svarið fleiri en 10 verður tekið meðaltal allra þeirra dæma sem þið reynið við. Öll dæmi gilda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

Munið að öll föll eiga að hafa notkunarlýsingu (notkun/fyrir/eftir).

Engin hjálpargögn eru leyfileg.

Hluti I: Bálkmótun o.fl.

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




```
(define (A ...)
   (define (B ...)
      (define (C ...)
         ...
```

```
        )
        ...
    )
    (define (D ...)
        ...
    )
    ...
)
```

EÐA / OR (jafngilt / equivalent)

```
(define (A ...)
    (define (D ...)
        ...
    )
    (define (B ...)
        (define (C ...)
            ...
        )
        ...
    )
    ...
)
```

(a) Kalla má á A úr B.

(b) Kalla má á A úr C.

(c) Kalla má á A úr D.

(d) Kalla má á B úr A.

(e) Kalla má á B úr C.

(f) Kalla má á B úr D.

(g) Kalla má á C úr A.

(h) Kalla má á C úr B.

(i) Kalla má á C úr D.

(j) Kalla má á D úr A.

(k) Kalla má á D úr B.

(l) Kalla má á D úr C.

(m) Í A má nota staðværar breytur úr B.

(n) Í A má nota staðværar breytur úr C.

(o) Í A má nota staðværar breytur úr D.

(p) Í B má nota staðværar breytur úr A.

(q) Í B má nota staðværar breytur úr C.

(r) Í B má nota staðværar breytur úr D.

(s) Í C má nota staðværar breytur úr A.

(t) Í C má nota staðværar breytur úr B.

(u) Í C má nota staðværar breytur úr D.

(v) Í D má nota staðværar breytur úr A.

(w) Í D má nota staðværar breytur úr B.

(x) Í D má nota staðværar breytur úr C.

2. (a) Hvað er stýrihlekkur (einnig kallað *control link, dynamic link*) og hver er tilgangur hans? Hver konar keðju færðu ef þú gengur í gegnum þau minnissvæði sem stýrihlekkir tengja saman í keðju? Hvaða minnissvæði og hvers konar er fremst í keðjunni, næstfremst og aftast?

   (b) Hvað er aðgangshlekkur (einnig kallað tengihlekkur, *access link, static link*) og hver er tilgangur hans? Hver konar keðju færðu ef þú gengur í gegnum þá hlekki sem aðgangshlekkir tengja saman í keðju? Hvaða hlekkur og hvers konar er fremst í keðjunni, næstfremst og aftast?

3. Hverjar eftirfarandi fullyrðinga um lokanir (*closure*) eru sannar? Tvö röng svör valda því að ekkert fæst fyrir dæmið.

   (a) Lokanir innihalda tengihlekk (*aðgangshlekk, access link, static link*).

   (b) Lokanir innihalda stýrihlekk (*control link, dynamic link*).

   (c) Lokanir innihalda vakningarfærslu.

   (d) Lokanir innihalda bendi á vakningarfærslu.

   (e) Lokanir innihalda bendi á stef eða fall.

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

   (g) Lokanir eru sveigjanlegri í notkun í Scheme en í Pascal vegna þess 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) C++ hefur lokanir.

(k) Morpho hefur lokanir.

(l) Haskell hefur lokanir.

(m) Java hefur lokanir.

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

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

int i,a[10];
for( i=0 ; i!=10 ; i++ ) a[i]=i;
f(a[a[0]],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.

5. Skrifið halaendurkvæmt[1] fall í Scheme, Haskell, CAML eða Morpho, sem tekur eina heiltölu $n \geq 0$ sem viðfang og skilar heiltölulistanum (2^n 2^{n-1} ... 4 2 1). Ef fallið er ekki halaendurkvæmt fæst aðeins hálft fyrir dæmið.

6. Forritið straum í Scheme, Haskell eða Morpho sem inniheldur óendanlegu rununa $[1, 1, 2, 3, 5, \ldots]$ af Fibonacci tölunum.

7. 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.

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

8. Skrifið fall `s` í Scheme með eftirfarandi notkunarlýsingu:

   Notkun: `(s f x g)`

   Fyrir: `f` er tvíundarfall, `g` er einundarfall.

   Gildi: Fall `h` þ.a. segðin `(h z)` skilar sama gildi og segðin `(f x (g z))`.

   Skrifið einnig samsvarandi fall í CAML eða Haskell. Hvað er tagið á því falli?

Hluti III: Einingaforritun o.fl.

9. 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: $e < x < f$.
     Eftir: $g < y < h$.
   - Lýsing í B:

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

   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 og skal lýsa innbyrðis röð gildanna.

10. Skrifið hönnunarskjal fyrir einingu í Morpho, Java eða C++ sem útfærir hlaða (*stack*) gilda (eða hluta). Einingin skal vera nægilega fjölnota til að unnt sé að búa til hlaða 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 hlaðinn innihaldi aðeins gildi af því tagi sem til er ætlast.

    Munið upplýsingahuld. Aðeins þarf hönnunarskjal, enga útfærslu þarf.

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

    Munið upplýsingahuld.

Skrifið einnig fastayrðingu gagna og útfærið aðgerðina til að bæta við gildi í forgangsbiðröðina.

12. Skrifið hönnunarskjal fyrir fjölnota einingu í Morpho sem raðar lista gilda ef einhverri óþekktri gerð. Úr einingunni skal vera útflutt aðgerð til að raða og innfluttar skulu vera aðgerðir eða aðgerð til að bera gildi saman.

Hluti IV: Blandað efni

13. Íhugið ruslasöfnunaraðferðirnar *merkja og sópa* (*mark and sweep*) annars vegar og *afritunarsöfnun* (*stop and copy*) hins vegar. Berið saman þessar tvær aðferðir varðandi tímagróðann sem fæst af því að stækka minnið sem notað er undir kösina (*heap*). Er meiri gróði af slíku í annarri aðferðinni? Hví eða hví ekki?

14. Gefin er eftirfarandi BNF mállýsing (ϵ er tómi strengurinn):

```
<E> ::= <E> <OP> <E>
<E> ::= <E> *
<E> ::= ( <E> )
<E> ::= a
<E> ::= b
<OP> ::= +
<OP> ::= ϵ
```

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

(a) `a+b`

(b) `ab`

(c) `a*b`

(d) `*ab`

(e) `ab*`

(f) `(a)*b`

(g) `(a*)b`

(h) `(a*b)`

15. 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

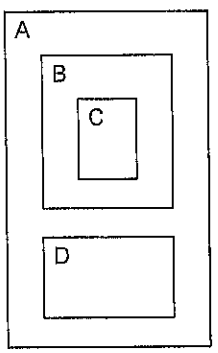
Sýnið þrjá strengi sem eru í báðum málunum og þrjá strengi sem eru í öðru málinu en ekki í hinu.

TÖL304G Programming Languages
Final Exam Autumn 2013

Answer 10 of the following questions and at least two from each of chapters I–III. If you answer more than 10 questions you should specify which answers should count toward your grade. If you do not specify such answers and answer more than 10 then the average will be taken from all your answer attempts. All questions count equally. **Write on the foremost white page in your solution book which solutions should count toward your grade.**
Remember that all functions should have a description (usage/pre/post).
No help materials are allowed.

Part I: Block-Structure etc.

1. Assume nesting in a block-structured programming language as shown in the figure and as the Scheme code segments show. Which of the following statements are then true and which are false? One wrong answer causes the grade for the question to be zero.




```
(define (A ...)
   (define (B ...)
      (define (C ...)
         ...
```

```
        )
        ...
    )
    (define (D ...)
        ...
    )
    ...
)
```

EÐA / OR (jafngilt / equivalent)

```
(define (A ...)
    (define (D ...)
        ...
    )
    (define (B ...)
        (define (C ...)
            ...
        )
        ...
    )
    ...
)
```

(a) A may be called from B.

(b) A may be called from C.

(c) A may be called from D.

(d) B may be called from A.

(e) B may be called from C.

(f) B may be called from D.

(g) C may be called from A.

(h) C may be called from B.

(i) C may be called from D.

(j) D may be called from A.

(k) D may be called from B.

(l) D may be called from C.

(m) In A you can use local variables in B.

(n) In A you can use local variables in C.

(o) In A you can use local variables in D.

(p) In B you can use local variables in A.

(q) In B you can use local variables in C.

(r) In B you can use local variables in D.

(s) In C you can use local variables in A.

(t) In C you can use local variables in B.

(u) In C you can use local variables in D.

(v) In D you can use local variables in A.

(w) In D you can use local variables in B.

(x) In D you can use local variables in C.

2. (a) What is a control link (also called *dynamic link*, *stýrihlekkur*) and whatr is its purpose? What kind of chain do you get if you traverse the memory areas chained together by the control links one by one? What memory area and what kind of memory area is at the beginning of the chain, next to the beginning of the chain and at the end of the chain?

   (b) What is an access link (also called *static link*, *öhtengihlekkur*, aðgangshlekkur) *and what is its purpose? What kind of chain do you get if you traverse the memory areas chained together by the access links one by one? What memory area and what kind of memory area is at the beginning of the chain, next to the beginning of the chain and at the end of the chain?*

3. Which of the following statements on closures are true? Two wrong answers cause the grade for the question to be zero.

   (a) Closures contain an access link (static link).

   (b) Closures contain a control link (dynamic link).

   (c) Closures contain an activation record.

   (d) Closures contain a pointer to an activation records.

   (e) Closures contain a pointer to a procedure or function.

   (f) Closures are more flexible in Scheme than in Pascal because in Scheme we can store closures on the heap.

(g) Closures are more flexible in Scheme than in Pascal because in Scheme we can store activation records on the heap.

(h) In Scheme a closure can be a return value from a function but not in Standard Pascal.

(i) CAML has closures.

(j) C++ has closures.

(k) Morpho has closures.

(l) Haskell has closures.

(m) Java has closures.

4. The following program text is in some imagined programming language. (Ask if you feel the syntax or semantics is not obvious.)

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

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

What does this program write (five values each time) if the arguments are

- passed by value?
- passed by reference?
- passed by name?

Part II: List Processing etc.

5. Write a tail-recursive[1] function in Scheme, Haskell, CAML or Morpho that takes one integer argument, $n \geq 0$, and returns the integer list (2^n 2^{n-1} ... 4 2 1). If the function is not tail-recursive only half the points are given for the answer.

[1] It is enough that the implementation use a tail recursive help function to restrict the depth of the recursion. You may not do a loop implementation in Morpho.

6. Program a stream in Scheme, Haskell or Morpho that contains the infinite series $[1, 1, 2, 3, 5, \ldots]$ of Fibonacci numbers.

7. Write two functions in Scheme, CAML, Morpho or Haskell that are traditional implementations of insert-left and insert-right, i.e. functions that take three arguments and apply a binary operation from the right or from the left.

8. Write a function `s` in Scheme with the following description:

   Usage: `(s f x g)`

   Pre: `f` is a binary function, `g` is a unary function.

   Value: A function `h` such that the expression `(h z)` returns the same value as the expression `(f x (g z))`.

   Also write a similar function in CAML or Haskell. What is the type of that function?

Part III: Modular Programming etc.

9. Assume the existence of classes A and B in some object-oriented programming language and that B is a subclass of A and that both classes have a message t with the following descriptions in A and B.

   - Description in A:

     Usage: y = z.t(x);
     Pre: $e < x < f$.
     Post: $g < y < h$.

   - Description in B:

     Usage: y = z.t(x);
     Pre: $a \leq x \leq b$.
     Post: $c \leq y \leq d$.

   What does the relationship between a, b, c, d, e, f, g, and h in order for the logical relationship between the classes A and B to be correct? The relationship you specify must be necessary and sufficient and should describe the relative ordering of the values.

10. Write a design document for a module in Morpho, Java or C++ that implements a stack of values (or objects). The module should be polymorphic (fjölnota) enough so that you can make a stack of any values that fulfill reasonable constraints, but the module should also use the typing constraints

of the programming language to ensure as far as reasonable that the stack only contains values of the relevant type.

Remember information hiding. Only a design document is needed, no implementation is needed.

11. Write a design document for a polymorphic module for priority queues in Java, C++ or Morpho.

    Remember information hiding.

    Also write a data invariant (fastayrðing gagna) and program the method to add a value to the priority queue.

12. Write a design document for a polymorphic module in Morpho that implements sorting of lists of values of unknown sort. Exported from the module should be an operation for sorting imports should be or more operations to compare values.

Part IV: Miscellaneous

13. Consider the garbage collection methods *mark and sweep* on one hand and *stop and copy* on the other hand. Compare these two methods as regards the time savings resulting from increasing the memory used for the heap. Are there more savings to be had from one of these methods? Why or why not?

14. Consider the following BNF grammar (ϵ is the empty string):

```
<E> ::= <E> <OP> <E>
<E> ::= <E> *
<E> ::= ( <E> )
<E> ::= a
<E> ::= b
<OP> ::= +
<OP> ::= ϵ
```

    Which of the following strings are in the language (one wrong answer gives grade zero)?

    (a) `a+b`

    (b) `ab`

    (c) `a*b`

(d) `*ab`

(e) `ab*`

(f) `(a)*b`

(g) `(a*)b`

(h) `(a*b)`

15. Consider the languages defines on one hand by the regular expression $(x(x(x(xy)^*y)^*y)^*y)^*$ and on the other hand by the BNF grammar below. Note that ϵ stands for the empty string.

    S ::= x <S> y <S> | ϵ

    Show three streing that are in both languages and three strings that are in one language but not the other.