TÖL304G Forritunarmál

Prófdagur og tími: 17.12.2014 13:30-16:30

Prófstaður:
Aðalbygging - A225 (fjöldi: 6)
Aðalbygging - A229 (fjöldi: 3)
VR-2 - V138 (fjöldi: 18)
VR-2 - V152 (fjöldi: 32)
VR-2 - V155 (fjöldi: 21)
VR-2 - V156 (fjöldi: 21)
VR-2 - V258 (fjöldi: 21)
VR-2 - V261 (fjöldi: 30)

HÁSKÓLI ÍSLANDS

Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild

Skriflegt próf

Skráðir til prófs: 152

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

Kennslumisseri: Haust 2014

Úrlausnir skulu merktar með nafni

Prófbók/svarblöð:
Línustrikuð prófbók

Hjálpargögn:
Engin hjálpargögn eru leyfileg.

Önnur fyrirmæli: Prófspurningar eru bæði á ensku og íslensku

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

Einkunnir skulu skráðar í Uglu eigi síðar en 07.01.2015.

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 2014
TÖL304G Programming Languages
Final Exam Autumn 2014

Svarið **fyrstu fjórum dæmunum** og svarið **tíu dæmum í heild**. 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.

Answer the **first four questions** and answer **a total of ten questions**. 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.

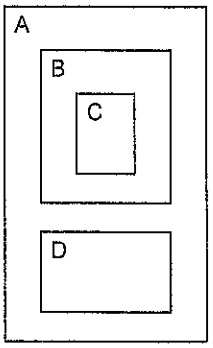
Hluti I: Skylduspurningar
Part I: Mandatory Questions

1. Sýnið BNF, EBNF eða málrit fyrir mál λ-reiknisegða með svigum, breytunöfnunum x, y og z, fallsbeitingu NM þar sem N og M eru λ-reiknisegðir, og fallsskilgreiningum $\lambda a.N$ þar sem a er breytunafn og N er λ-reiknisegð. Dæmi um löglegar formúlur eru þá t.d. $\lambda x.xy$, $x(xy)$, xyz og $(\lambda x.yx)z$, en ekki t.d. a eða $\lambda xy.z$.

   Show BNF, EBNF or syntax diagrams for a language og λ-expressions with parentheses, the variable names x, y and z, function applications NM where N and M are λ-expressions, and function definitions $\lambda a.N$ where a is a variable name and N is a λ-expression. Examples of legal expressions are $\lambda x.xy$, $x(xy)$, xyz and $(\lambda x.yx)z$, but not, for example a or $\lambda xy.z$.

2. 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.

   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) 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.

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.

3. Forritið fall í Scheme, CAML, Haskell eða Morpho sem tekur gildi u, tvíundaraðgerð $+$ og lista $x = [x_1, \ldots, x_n]$ og skilar gildinu $u + x_1 + \ldots + x_n$, reiknað frá vinstri til hægri.

   Forritið einnig fall í Scheme, CAML, Haskell eða Morpho sem tekur gildi u, tvíundaraðgerð $+$ og lista $x = [x_1, \ldots, x_n]$ og skilar gildinu $x_1 + \ldots + x_n + u$, reiknað frá hægri til vinstri.

   Að minnsta kosti annað fallið skal vera halaendurkvæmt.

Aðeins má nota grunnaðgerðir svo sem car, cdr og null? í Scheme.

Program a function in Scheme, CAML, Haskell or Morpho that takes a value u, a binary operation $+$ and a list $x = [x_1, \ldots, x_n]$ and returns the value $u + x_1 + \ldots + x_n$, computed from left to right.

Also program a function in Scheme, CAML, Haskell or Morpho that takes a value u, a binary operation $+$ and a list $x = [x_1, \ldots, x_n]$ and returns the value $x_1 + \ldots + x_n + u$, computed from right to left.

At least one of the functions must be tail-recursive. Only fundamental operations such as car, cdr, and null? in Scheme may be used.

4. 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.

   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.

Hluti II: Bálkmótun o.fl.
Part II: Block-Structure etc.

5. 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.

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.

6. 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 með því að tiltaka eitthvað sem hægt er að gera í öðru forritunarmálinu með vakningarfærslur í kös, sem ekki væri hægt að gera ef vakningarfærslurnar væru ekki í kös.

   Answer all of the following:

   a) What information is stored in activation records?

b) Which of these information items are only in block-structured programming languages and why?

c) Name two programming languages where activation records are usually stored on the stack and two where the activation records are usually on the heap.

d) Show why, in the latter two programming languages, it is necessary for the activation records to be stored on the heap by describing something that can be done in either of the programming languages with activation records on the heap that would not be possible if the activation records were not on the heap.

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

   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];
```

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

- gildisviðföng?
- tilvísunarviðföng?
- nafnviðföng?

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

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

Hluti III: Listavinnsla o.fl.
Part III: List Processing etc.

8. 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.

   Write a function in Scheme, CAML, Morpho or Haskell that takes one argument that is a list of lists of floating point numbers between 0 and 1 and returns a number that is the lowest maximum of the inner lists, i.e. the lowest among the numbers that result from taking the maximum of each inner list.

You may assume that the maximum of the empty set is 0 and the minimum of the empty set is 1.

9. Skrifið fall biApplySum í Scheme, CAML, Morpho eða Haskell sem tekur þrjú viðföng, lista af tvíundarföllum og tvo jafn langa lista af gildum og skilar fleytitölusummu skilagildanna (sem verða að vera fleytitölur) sem út koma þegar föllunum úr fyrsta listanum er beitt eitt af öðru á gildin úr hinum listunum. Segið einnig til um tagið á fallinu samkvæmt tögunarreglum CAML eða Haskell (hvort sem þið skrifið fallið í CAML eða Haskell eður ei). Dæmi um notkun í Scheme væri (biApplySum (list + /) (list 1 2) (list 3 4)), sem skila ætti gildinu $(1 + 3) + (2/4) = 4.0 + 0.5 = 4.5$.

   Write a function biApplySum in Scheme, CAML, Morpho or Haskell that takes three arguments, a list of binary functions and two equally long lists of values and returns the floating point sum of the results (that have to be floating point numbers) that result from applying the functions in the first list, one by one, to the corresponding values in the other lists. Also state the type of the function according to the typing rules of CAML or Haskell (whether or not you write the function on CAML or Haskell or not). An example of using the function in Scheme would be (biApplySum (list + /) (list 1 2) (list 3 4)), which should return the value $(1 + 3) + (2/4) = 4.0 + 0.5 = 4.5$.

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

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

listanum $[n, n-1, \ldots, 1]$. Ekki má nota hliðarverkanir og ef þið notið Morpho mega breytur aldrei fá ný gildi eftir að þær eru frumstilltar.

Write a tail recursive[2] function in Scheme, CAML, Haskell or Morpho, that takes as argument one integer $n \geq 0$ and returns the list $[n, n-1, \ldots, 1]$. You may not us side effects and if you use Morpho you may never give a variable a new value after it is initialized.

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

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

12. 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?

    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)).

[2]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.

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

Hluti IV: Einingaforritun o.fl.
Part IV: Modular Programming etc.

13. 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.

    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.

14. 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.

Write a design document for a polymorphic module in Mopho 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.

Hluti V: Blandað efni
Part V: Miscellaneous

15. 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?

    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?

16. 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 '(((())))'. Draw a finite state machine (deterministic or nondeterministic) for the language over the alphabet $\{(,)\}$ that contains the strings that have balanced parentheses and the nesting depth

is at most 3. The strings '(())' and '()(((()))' should be in the language, but not '(((())))'.

17. Hvað er sniðmengi málanna sem hafa eftirfarandi BNF málýsingar? Sýnið einnig þrjá strengi sem eru í sniðmenginu og þrjá strengi sem eru í sammenginu en ekki í sniðmenginu.

    What is the set intersection of the languages with the following BNF grammars? Also show three string that are in the intersection and three strings that are in the union but not in the intersection.

    Mál/Language A:

    ```
    <A> ::= x <A> z
    <A> ::= <B>
    <B> ::= y <B>ϵ
    <B> ::= ϵ
    ```

    Mál/Language B:

    ```
    <D> ::= x <D>
    <D> ::= <E>
    <E> ::= y <E> z
    <E> ::= ϵ
    ```