TÖL304G
Forritunarmál
Verkefnablað 6

Snorri Agnarsson

26. september 2022

Verkefni — Exercises

Munið að föll sem skilað er, þar með talið hjálparföll, hafa skýra og rétta lýsingu með „Notkun: ...“, „Fyrir: ...“ og „Gildi: ...“. Takið eftir að í sumum tilfellum þurfa forskilyrði að innihalda lýsingar á sama sniði fyrir viðföng sem eru föll, og svipað gildir í eftirskilyrði (þ.e. „Gildi: ...“) fyrir gildi sem eru föll.

Remember that all functions turned in, including helper functions, should have a clear and correct description with "Use: ...", "Pre: ...", and "Value: ...". Note that in some instances preconditions need to contain similar descriptions for arguments, and similarly for the postcondition (i.e. "Value: ...") for values that are functions.

Mikilvægt er að – It is important that:

- Allar lausnir séu vel sniðsettar og *með réttri innfellingu*.

  All solutions are well formatted and *correctly indented*.

- Öll **föll** hafi skýra lýsingu með „Notkun:“, „Fyrir:“ og annaðhvort „Eftir:“ eða „Gildi:“, þ.e. forskilyrði og eftirskilyrði. Sé lýsingin gefin fyrirfram skal nota hana, en athugið að hjálparföll sem þið finnið sjálf upp þurfa sína lýsingu. Athugið samt að þótt gefin sé lýsing falls má skrifa aðra lýsingu ef forskilyrði er víkkað og eftirskilyrði er þrengt. Það kemur fyrir sð slíkt sé gagnlegt, til dæmis ef endurkvæm notkun fallsins krefst betra falls en upphaflega lýsingin krefst.

  All **functions** have a clear description with "Use:", "Pre:"and either "Post:or "Value:", i.e. preconditions and postconditions. If the description is given in the exercise definition, that should be used, but note that helper functions that you invent yourself need their descriptions. However, it should be noted that

even though a description is given, you can supply another description as long as the new description has a more relaxed precondition and a more restrictive postcondition. Sometimes this is useful, for example if recursive use requires a function with an improved description.

- Sérhver nemandi skal vinna sín einstaklingsverkefni einn og óháður öðrum nemendum. Samræður milli nemenda um verkefnin eru af hinu góða, en afritun einstaklingsverkefna er að sjálfsögðu óheimil.

  Each student should solve their individual exercises be themself, independently of other students. Conversations between studentt are good, but copying of individual solutions is strictly forbidden.

Hópverkefni — Group Assignments

Fyrir þessi verkefni skuluð þið sækja skrána `permutations.ml-beinagrind` úr Canvas og vista hana hjá ykkur undir nafninu `permutations.ml`. Gerið síðan viðeigandi viðbætur og sýnið forritstextannn ásamt lýsingum fyrir föllin sem þið klárið.

For this assignment you should fetch the file `permutations.ml-beinagrind` from Canvas and file it on your computer under the name `permutations.ml`. Then do the appropriate additions and show the source code with descriptions for the functions you finish.

1. Skrifið fall mapreduce í CAML, þar sem kallið mapreduce f op u x tekur fall f, tvíundaraðgerð op, lista x = $[x_1; ...; x_n]$ og gildi u og skilar gildinu $u \oplus f(x_1) \oplus \ldots \oplus f(x_n)$, þar sem $x \oplus y = (op\ x\ y)$. Fallið skal vera halaendurkvæmt og skal reikna frá vinstri til hægri.

   Write a function mapreduce in CAML, where the call mapreduce f op x u takes a function f, a binary operation op, a list x = $[x_1; ...; x_n]$, and a value u and returns the value $u \oplus f(x_1) \oplus \ldots \oplus f(x_n)$, where $x \oplus y = (op\ x\ y)$. The function must be tail recursive and should compute from left to right.

   Sýnið eftirfarandi prófun á fallinu — Show the following test of the function:

   ```
   let
     inc x = x+1
   and
     add x y = x+y
   in
     mapreduce inc add 0 [0;1;2;3;4];;
   ```

2. Klárið að forrita fallið `fromTo`. Munið að sýna prófanir.

   Finish programming the function `fromTo`. Remember to show tests.

3. Klárið að forrita fallið `insertAt`. Munið að sýna prófanir.

   Finish programming the function `insertAt`. Remember to show tests.

4. Klárið að forrita fallið `extendPermutation`. Munið að sýna prófanir.

   Finish programming the function `extendPermutation`. Remember to show tests.

5. Klárið að forrita fallið `length`. Munið að sýna prófanir.

   Finish programming the function `length`. Remember to show tests.

6. Þegar þessu er öllu lokið, sýnið útkomuna úr segðinni — When this is done, show the result from the expression

   ```
   length (permutations 6);;
   ```

   sem er aftast í skránni ykkar — which is at the end of your file.

Einstaklingsverkefni — Individual Assignments

Klárið að forrita eftirfarandi CAML fall — Finish programming the following CAML function.

```
(*
Use:    powerList i j
Pre:    i, j are integers.
Value:  The list of lists [x1;x2;...;xN]
        containing all the sublists of the
    ascending list of integers k such
    that i<=k<=j. If i<=j the result is
    all the sublists of [i;i+1;...;j].
    If i>j the result is [[]].

Notkun: powerList i j
Fyrir:  i, j eru heiltölur.
Gildi:  Listi lista [x1;x2;...;xN] sem
        inniheldur alla undirlista lista
        allra heiltalna k þ.a. i<=k<=j,
        í vaxandi röð. Ef i<=j þá er
        útkoman allir undirlistar listans
        [i;i+1;...;j].
        Ef i>j þá er útkoman [[]].
*)
```

```
let rec powerList i j =
    ...
```

Þið megið nota hvaða innbyggð föll í CAML sem ykkur hentar. Athugið að tvíundaraðgerðin `@` skeytir saman tveimur listum. Ef `n` er heiltala og `x` er heiltalnalisti þá gefur segðin `n::x` listann þar sem `n` er skeytt framan á `x` (eins og `cons` í Scheme). Munið að sýna prófanir.

You may use whatever built-in function in CAML that you wish. Note that the binary operation `@` appends two lists. If `n` is an integer and `x` is a list of integers then the expression `n::x` gives the list where `n` is prepended to `x` (like `cons` in Scheme). Remember to show tests.