TÖL304G
Forritunarmál
Lausnir 5

Snorri Agnarsson

27. september 2020

Verkefni — Exercises

Hópverkefni — Group Assignments

1. Skrifið halaendurkvæmt fall `modpow` sem tekur þrjú viðföng, p, q og r sem eru heiltölur og geta verið mörg hundruð stafa tölur og skilar afgangnum þegar r er deilt í p^q. Athugið að það dugar að allar milliniðurstöður séu smækkaðar í jafngildar tölur módúló r.

   Þið megið gera ráð fyrir að $q > 0$ og að $0 \leq p < r$. Verið varkár í lýsingum fallsins og hjálparfallsins og sjáið til þess að öll endurkvæm köll séu lögleg miðað við lýsingar.

   Föll svo sem `modpow` eru kjarninn í ýmsum dulritunaraðferðum svo sem Diffie-Hellman og RSA, sem mikið eru notaðar í tölvusamskiptum.

   Scheme föllin `quotient` og `remainder` verða væntanlega gagnleg í þessu verkefni.

   Sýnið útkomurnar úr `(modpow 123 1234567890 12345678901)` og `(modpow 2 10 10000)`.

   Write a tail recursive function `modpow` that takes three arguments, p, q, and r that are integers, and may have hundreds of digits, and returns the remainder when r is divided into p^q. Note that it is enough that all intermediate results are reduced to smaller numbers that are equal modulo r.

You may assume that $q > 0$, and that $0 \leq p < r$. Be careful in your descriptions of the function and helper function and make sure thet all recursive calls are legal considering the descriptions.

Functions such as `modpow` are the basis for various encryption methods such as Diffie-Hellman and RSA that are in wide use in computer communication.

The Scheme functions `quotient` and `remainder` will presumably be useful in this exercise.

Show the results from `(modpow 123 1234567890 12345678901)` and `(modpow 2 10 10000)`.

Svar/Answer:

```
;; Notkun: (modpow p q r)
;; Fyrir:  p, q og r eru heiltölur,
;;         0 <= p < r, q >= 0.
;; Gildi:  Heiltalan (p^q)%r.

;; Use:    (modpow p q r)
;; Pre:    p, q, and r are integers,
;;         0 <= p < r, q >= 0.
;; Value:  The integer (p^q)%r.
(define (modpow p q r)
    ;; Notkun: (hjalp p q s)
    ;; Fyrir:  p, q og s eru heiltölur,
    ;;         0 <= p,s < r, q >= 0.
    ;; Gildi:  Heiltalan (s*(p^q))%r.

    ;; Use:    (hjalp p q s)
    ;; Pre:    p, q, and s are integers,
    ;;         0 <= p < r, q >= 0.
    ;; Value:  The integer (s*(p^q))%r.
    (define (hjalp p q s)
        (if (= q 0)
            s
            (if (= (remainder q 2) 0)
                (hjalp (remainder (* p p) r)
                       (quotient q 2)
                       s
                )
                (hjalp (remainder (* p p) r)
                       (quotient q 2)
                       (remainder (* p s) r)
```

```
                    )
                )
            )
        )
        (hjalp p q 1)
)

(modpow 123 1234567890 12345678901)
;; skilar/returns 10385213685

(modpow 2 10 10000)
;; skilar/returns 1024
```

2. Skrifið fall `cornerstream` með eftirfarandi lýsingu.

   Write a function `cornerstream` with the following description.

```
;; Notkun: (cornerstream s n)
;; Fyrir:  s er óendanlegur straumur óendanlegra
;;         strauma,
;;         s=[[x11 x12 ...],[x21 x22 ...] ...].
;;         n er heiltala, n>=0.
;; Gildi:  Listinn
;;         ((x11 x12 ... x1n)
;;          (x21 x22 ... x2n)
;;          ...
;;          (xn1 xn2 ... xnn)
;;         )

;; Notkun: (cornerstream s n)
;; Fyrir:  s is an infinite stream of
;;         infinite streams,
;;         s=[[x11 x12 ...],[x21 x22 ...] ...].
;;         n isan integer, n>=0.
;; Gildi:  The list
;;         ((x11 x12 ... x1n)
;;          (x21 x22 ... x2n)
;;          ...
;;          (xn1 xn2 ... xnn)
;;         )
```

Svar/Answer:

```
;; Notkun: (cornerstream s n)
;; Fyrir:  s er óendanlegur straumur óendanlegra
;;         strauma,
;;         s=[[x11 x12 ...],[x21 x22 ...] ...].
;;         n er heiltala, n>=0.
;; Gildi:  Listinn
;;         ((x11 x12 ... x1n)
;;          (x21 x22 ... x2n)
;;          ...
;;          (xn1 xn2 ... xnn)
;;         )

;; Notkun: (cornerstream s n)
;; Fyrir:  s is an infinite stream of
;;         infinite streams,
;;         s=[[x11 x12 ...],[x21 x22 ...] ...].
;;         n isan integer, n>=0.
;; Gildi:  The list
;;         ((x11 x12 ... x1n)
;;          (x21 x22 ... x2n)
;;          ...
;;          (xn1 xn2 ... xnn)
;;         )
(define (cornerstream s n)
    (map (lambda (x) (stream-list x n))
         (stream-list s n)
    )
)

;; Prófun/Tests:
(define heilsquare (cons-stream heil heilsquare))
(cornerstream heilsquare 4)
;; skilar/returns
;; ((1 2 3 4) (1 2 3 4) (1 2 3 4) (1 2 3 4))
```

3. Skrifið fall `mulstreams` með eftirfarandi lýsingu.

   Write a function `mulstreams` with the following description.

```
;; Notkun: (mulstreams x y)
;; Fyrir:  x og y eru óendanlegir straumar talna,
;;         x=[x1 x2 x3 ...].
```

```
;;          y=[y1 y2 y3 ...].
;; Gildi:   Óendanlegur straumur óendanlegra strauma
;;          talna sem er
;;          [[x1*y1 x2*y1 x3*y1 ...]
;;           [x1*y2 x2*y2 x3*y2 ...]
;;           [x1*y3 x2*y3 x3*y3 ...]
;;           .
;;           .
;;           .
;;          ]

;; Use:     (mulstreams x y)
;; Pre:     x and y are infinite streams of numbers,
;;          x=[x1 x2 x3 ...].
;;          y=[y1 y2 y3 ...].
;; Value:   An infinite stream of infinite streams
;;          of numbers, namely
;;          [[x1*y1 x2*y1 x3*y1 ...]
;;           [x1*y2 x2*y2 x3*y2 ...]
;;           [x1*y3 x2*y3 x3*y3 ...]
;;           .
;;           .
;;           .
;;          ]
```

Prófið fallið `mulstreams` með segðinni `(mulstreams heil heil)` og sýnið hluta af útkomunni. Útkoman er óendanleg margföldunartafla þannig að vonlaust er að sjá hana alla, en athugið að nota má fallið `cornerstream` að ofan til a sjá hluta af henni.

Athugið að þægileg aðferð til að þróa föll sem byggð eru á straumaföllunum er að sækja forritstextann fyrir strauma úr Canvas og bæta ykkar falli aftast í skrána.

Test the function `mulstreams` with the expression `(mulstreams heil heil)` and show part of the result. The result is an infinite multiplication table so we can not hope to see all of it, but note that you can use the function `cornerstream` above to see a part of it.

Note that a convenient method for developing functions based on the stream functions is to fetch the source code for streams from Canvas and add your functions to the end of the file.

Svar/Answer:

```
;; Notkun: (mulstreams x y)
;; Fyrir:  x og y eru óendanlegir straumar talna,
;;         x=[x1 x2 x3 ...].
;;         y=[y1 y2 y3 ...].
;; Gildi:  Óendanlegur straumur óendanlegra strauma
;;         talna sem er
;;         [[x1*y1 x2*y1 x3*y1 ...]
;;          [x1*y2 x2*y2 x3*y2 ...]
;;          [x1*y3 x2*y3 x3*y3 ...]
;;           .
;;           .
;;           .
;;         ]

;; Use:    (mulstreams x y)
;; Pre:    x and y are infinite streams of numbers,
;;         x=[x1 x2 x3 ...].
;;         y=[y1 y2 y3 ...].
;; Value:  An infinite stream of infinite streams
;;         of numbers, namely
;;         [[x1*y1 x2*y1 x3*y1 ...]
;;          [x1*y2 x2*y2 x3*y2 ...]
;;          [x1*y3 x2*y3 x3*y3 ...]
;;           .
;;           .
;;           .
;;         ]
(define (mulstreams x y)
    (cons-stream
        (stream-map
            (lambda (z) (* z (stream-car y)))
            x
        )
        (mulstreams x (stream-cdr y))
    )
)

;; Prófun/Testing:
;; (cornerstream (mulstreams heil heil) 3)
;; skilar/returns:
;; ((1 2 3) (2 4 6) (3 6 9))
```

4. Skrifið fall `powerlist` sem hefur eftirfarandi lýsingu.

   Write a function `powerlist` that has the following description.

```
;; Notkun: (powerlist n)
;; Fyrir:  n er heiltala, n>=0.
;; Gildi:  Listinn (y1 y2 y3 ...)
;;         sem inniheldur alla lista sem
;;         hægt er að smíða með því að taka
;;         núll eða fleiri gildi úr {1,...,n}
;;         og skeyta þeim saman í lista í
;;         minnkandi röð.

;; Use:    (powerlist n)
;; Pre:    n is an integer, n>=0.
;; Value:  The list (y1 y2 y3 ...)
;;         that contains all lists that can be
;;         constructed by taking zero or more
;;         values from {1,...,n}
;;         and concatenating them in a list in
;;         descending order.
```

   Athugið að þetta er svipað og að reikna veldismengi mengis. Röð listanna í útkomulistanum er valfrjáls. Til dæmis myndi `(powerlist 0)` skila listanum `(())` (sem samsvarar því að veldismengið af tómamenginu er mengið sem inniheldur aðeins tómamengið. Segðin `(powerlist 1)` gæti skilað listanum `(() (1))` eða listanum `((1) ())`. Segðin `(powerlist 2)` gæti skilað listanum `(() (1) (2) (2 1))`.

   Fjöldi staka (innri lista) í útkomulistanum ætti ávallt að vera 2^n.

   Innbyggðu föllin `map` og `append` eru gagnleg hér. Þetta verkefni er hægt að leysa á tiltölulega einfaldan hátt í Scheme.

   Notice that this is similar to computing the powerset of a set. The order of the lists in the result is arbitrary. For example, the expression `(powerlist 0)` should return the list `(())` (which corresponds to the fact that the powerset of the empty set is the set that contains only the empty set. The expression `(powerlist 1)` could return the list `(() (1))` or the list `((1) ())`. The expression `(powerlist 2)` could return the list `(() (1) (2) (2 1))`.

   The number of inner lists in the result list should always be 2^n.

   The built-in functions `map` and `append` are useful here. This problem can be solved in a relatively simple manner in Scheme.

   Svar/Answer:

```
;; Notkun: (powerlist n)
;; Fyrir:  n er heiltala, n>=0.
;; Gildi:  Listinn (y1 y2 y3 ...)
;;         sem inniheldur alla lista sem
;;         hægt er að smíða með því að taka
;;         núll eða fleiri gildi úr {1,...,n}
;;         og skeyta þeim saman í lista í
;;         minnkandi röð.

;; Use:    (powerlist n)
;; Pre:    n is an integer, n>=0.
;; Value:  The list (y1 y2 y3 ...)
;;         that contains all lists that can be
;;         constructed by taking zero or more
;;         values from {1,...,n}
;;         and concatenating them in a list in
;;         descending order.
(define (powerlist n)
    (if (= n 0)
        '(())
        (let ((pl (powerlist (- n 1))))
            (append
                pl
                (map (lambda (x) (cons n x))
                     pl
                )
            )
        )
    )
)

;; Eða/Or:
(define (powerlist n)
    (if (= n 0)
        '(())
        (append (powerlist (- n 1))
                (map (lambda (x) (cons n x))
                     (powerlist (- n 1))
                )
        )
    )
)
```

```
;; Prófun/Testing:

(powerlist 0)
(powerlist 1)
(powerlist 2)
(powerlist 3)
;; skilar/returns:
;; (())
;; (() (1))
;; (() (1) (2) (2 1))
;; (() (1) (2) (2 1) (3) (3 1) (3 2) (3 2 1))
```

Einstaklingsverkefni — Individual Assignments

1. Skrifið endurkvæmt Scheme fall sem samsvarar Dafny fallinu `RealPow_recursive` hér[1]. Skylda er að sjá til þess að dýpt endurkvæmni sé í mesta lagi í hlutfalli við $\log_2(x)$, alls ekki í hlutfalli við x, þegar reiknað er z^x.

   Þið munuð væntanlega vilja nota Scheme föllin `remainder` og `quotient`. Fyrir heiltölur `x` og `y` þannig að `x,y>0` gildir að `(remainder x y)` skilar afgangnum þegar `x` er deilt með `y`, og `(quotient x y)` skilar útkomunni úr heiltöludeilingu á `x` með `y`. Prófið þessi föll í Scheme til að sjá hverju þau skila og berið `quotient` saman við `(/ x y)`.

   Lýsing fallsins skal vera næganlega skýr til að glöggur lesandi geti sannfært sig um að virknin sé sönnuð án þess að bæta þurfi við forskilyrðum eða eftirskilyrðum titl þess að sönnunin gangi upp. Þið megið reikna með því að glöggur lesandi viti að $\left(x^2\right)^z = x^{2z}$.

   Prófið fallið með því að hefja töluna $1 + 10^{-10}$ í veldið 10^{10} og sýnið hvernig þið prófið og útkomuna úr prófinu. Þið skuluð sjá til þess að talan sem hafin er í veldi sé fleytitala frekar en ræð tala.

   Write a recursive Scheme function that corresponds to the Dafny function `RealPow_recursive` here[2]. You must ensure that the recursion depth is at most proportional to $\log_2(x)$, definitely not proportional to x, when computing z^x.

   You will presumably want to use the Scheme functions `remainder` and `quotient`. For integers `x` and `y` such that `x,y>0` we have that

[1] https://rise4fun.com/Dafny/SEz8

[2] https://rise4fun.com/Dafny/SEz8

`(remainder x y)` returns the remainder when `x` is divided by `y`, and `(quotient x y)` returns the integer quotient when `x` is divided by `y`. Try these functions in Scheme to see what they do and compare `quotient` to `(/ x y)`.

The description of the function must be clear enough so that a knowledgeable reader can convince himself that the functionality is correct without having to add preconditions or postconditions to comlete the proof. You may assume that the reader knows that $\left(x^2\right)^z = x^{2z}$.

Test the function by raising the number $1 + 10^{-10}$ to the power 10^{10} and show how you test and the result from the test. You should ensure that the number that is raised to a power is a floating point number rather than a rational number.

Svar/Answer:

```
;; Notkun: (pow x y)
;; Fyrir:  x er fleytitala,
;;         y er heiltala, y>=0.
;; Gildi:  x^y

;; Use:    (pow x y)
;; Pre:    x is a floating point number,
;;         y is an integer, y>=0.
;; Value:  x^y
(define (pow x y)
  (if (= y 0)
      1.0
      (if (= (remainder y 2) 0)
          (pow (* x x) (quotient y 2))
          (* x (pow (* x x) (quotient y 2)))
      )
  )
)

;; Prófun/Testing:
(pow 1.0000000001 10000000000)
;; skilar/returns: 2.7182820434752477
```

2. Skrifið fall `transpose-list` með eftirfarandi lýsingu.

   Write a function `transpose-list` with the following description.

```
;; Notkun: (transpose-list z)
;; Fyrir:  z er listi jafnlangra lista,
```

```
;;          z=((x11 x12 ... x1N)
;;             (x21 x22 ... x2N)
;;             (x31 x32 ... x3N)
;;             .
;;             .
;;             .
;;             (xM1 xM2 ... xMN)
;;           )
;; Gildi:  Listinn sem er byltingin
;;         (transpose) af z, þ.e.
;;         ((x11 x21 ... xM1)
;;          (x12 x22 ... xM2)
;;          (x13 x23 ... xM3)
;;          .
;;          .
;;          .
;;          (x1N x2N ... xMN)
;;         )

;; Use:    (transpose-list z)
;; Pre:    z is a list of lists that are
;;         all equal in lengths,
;;         z=((x11 x12 ... x1N)
;;            (x21 x22 ... x2N)
;;            (x31 x32 ... x3N)
;;            .
;;            .
;;            .
;;            (xM1 xM2 ... xMN)
;;          )
;; Value:  The list that is the transpose
;;         of z, i.e.
;;         ((x11 x21 ... xM1)
;;          (x12 x22 ... xM2)
;;          (x13 x23 ... xM3)
;;          .
;;          .
;;          .
;;          (x1N x2N ... xMN)
;;         )
```

Athugið að fallið þarf ekki að virka ef innri listarnir eru ekki jafnlangir því sam-

kvæmt forskilyrði á slíkt ekki að gerast. Ef innri listarnir eru tómir er eðlilegt að fallið skili tóma listanum. Sama gildir ef ytri listinn er tómur. Til hliðsjónar má kíkja á Dafny útfærslu á vefnum[3]. Munið að sýna prófanir.

Note that the function need not work if the inner lists are unequal in length because the precondition forbids this. If the inner lists are empty it is appropriate to return the empty list and the same holds if the outer list is empty. For reference you might look at a Dafny version on the web[4]. Remember to show tests.

Svar/Answer:

```
;; Notkun: (transpose-list z)
;; Fyrir:  z er listi jafnlangra lista,
;;         z=((x11 x12 ... x1N)
;;            (x21 x22 ... x2N)
;;            (x31 x32 ... x3N)
;;            .
;;            .
;;            .
;;            (xM1 xM2 ... xMN)
;;          )
;; Gildi:  Listinn sem er byltingin
;;         (transpose) af z, þ.e.
;;         ((x11 x21 ... xM1)
;;          (x12 x22 ... xM2)
;;          (x13 x23 ... xM3)
;;          .
;;          .
;;          .
;;          (x1N x2N ... xMN)
;;         )

;; Use:    (transpose-list z)
;; Pre:    z is a list of lists that are
;;         all equal in lengths,
;;         z=((x11 x12 ... x1N)
;;            (x21 x22 ... x2N)
;;            (x31 x32 ... x3N)
;;            .
;;            .
;;            .
;;            (xM1 xM2 ... xMN)
```

[3]https://rise4fun.com/Dafny/hpIj

[4]https://rise4fun.com/Dafny/hpIj

```
;;             )
;;  Value:  The list that is the transpose
;;          of z, i.e.
;;          ((x11 x21 ... xM1)
;;           (x12 x22 ... xM2)
;;           (x13 x23 ... xM3)
;;           .
;;           .
;;           .
;;           (x1N x2N ... xMN)
;;          )
(define (transpose-list z)
    (if (or (null? z) (null? (car z)))
        '()
        (cons (map car z)
              (transpose-list (map cdr z))
        )
    )
)

;; Prófun/Testing:
(define hh (cons-stream heil hh))
(cornerstream hh 3)
(transpose-list (cornerstream hh 3))
;; skilar/returns:
;; ((1 2 3) (1 2 3) (1 2 3))
;; ((1 1 1) (2 2 2) (3 3 3))
```