Programming Paradigms Fall 2022 — Problem Sets

by Nikolai Kudasov

September 22, 2022

1 Problem set №4

1. Using **explicit recursion**, implement the following functions:

   (a) Function `replicate` that creates a list with repeated values:

   ```
   (replicate 10 'a)
   ; '(a a a a a a a a a a)

   (replicate 3 '(1 . 2))
   ; '((1 . 2) (1 . 2) (1 . 2))
   ```

   (b) Function `split` that splits a given list into a pair of prefix of given size and remaining suffix.

   ```
   (split '(1 2 3 4 5) 2)
   ; '((1 2) . (3 4 5))

   (split '(a b c d) 4)
   ; '((a b c d) . ())

   (split '(a b c) 4)
   ; '((a b c) . ())

   (split '(a b c) 0)
   ; '(() . (a b c))
   ```

   (c) Function `chunks`, that splits a list into a list of chunks of given size:

   ```
   (chunks '(1 2 3 4 5) 2)
   ; '((1 2) (3 4) (5))

   (chunks '(a b c d e f) 3)
   ; '((a b c) (d e f))
   ```

   (d) Function `windows`, that produces a list of windows (sublists) of given size:

   ```
   (windows '(1 2 3 4 5) 2)
   ; '((1 2) (2 3) (3 4) (4 5))

   (windows '(a b c d e) 3)
   ; '((a b c) (b c d) (c d e))
   ```

2. Using higher-order functions (`apply`, `map`, `andmap`, `ormap`, `filter`, `foldl`), functions from the previous exercise, and **without explicit recursion**, implement the following functions:

   (a) Function `pairs`, that generates a list of all possible (unordered) pairs of elements from a given list:

   ```
   (pairs '(a b c d))
   ; '((a . b) (a . c) (a . d) (b . c) (b . d) (c . d))
   ```

   (b) Function `splits`, that generates all possible splits of a given list:

   ```
   (splits '(a b c))
   ; '(((a b c) . ()) ((a b) . (c)) ((a) . (b c)) (() . (b c)))
   ```

   (c) Function `max-product`, that finds two elements of the list that result in a maximum product:

   ```
   (max-product '(1 2 3 4 3 2 1))
   ; '(3 . 4)
   ```

   (d) Function `max-binary-op`, that finds two elements of the list that maximize a given binary function:

   ```
   (max-binary-op * '(1 2 3 4 3 2 1))
   ; '(3 . 4)

   (max-binary-op - '(1 2 3 4 3 2 1))
   ; '(4 . 1)
   ```

   (e) Function `combinations`, that generates a list of all possible (unordered) combinations of n elements from a given list:

   ```
   (combinations '(a b c d) 3)
   ; '((a b c) (a b d) (a c d) (b c d))
   ```

3. Implement the following functions, using `foldl`:

   (a) Function `max` that finds the maximum value:

   ```
   (max '(1 5 3 6 2 0))
   ; 6
   ```

   (b) Function `second-max` that finds the second maximum value:

   ```
   (second-max '(1 5 3 6 2 0))
   ; 5
   ```

   (c) Function `top-3` that returns a list of (at most) 3 maximum elements of a list (in any order):

   ```
   (top-3 '(5 3 6 2 8 1 0))
   ; '(5 6 8)
   ```

   (d) Function `group` that groups consecutive equal elements of a list into lists:

   ```
   (group '(a b b c c c b a a))
   ; '((a) (b b) (c c c) (b) (a a))
   ```

   (e) Function `cumulative-sums` that computes cumulative sums for all prefixes of a list:

   ```
   (cumulative-sums '(1 2 3 4 5))
   ; (0 1 3 6 10 15)
   ```