Exercise Sheet on APL
Parallel Functional Programming (PFP) 2017/2018
Version 1.06

Department of Computer Science, University of Copenhagen

2017-12-18

Introduction

These exercises aim at practising the use of APL with particular emphasis on the use of SOAC functions and the dfns[1] subset of APL. It should be possible to solve the exercises using either GNU APL or `tryapl.org`. Points are given according to the following table:

Exercise	1.1	1.2	1.3	2.1	2.2	3.1	4.1	Total
Points	10	10	10	10	20	30	10	100

1 Computing on Signals

Exercise 1.1 10 P.

Write a dyadic function `sumsq` that sums the squares of its two argument arrays.

```
     a ← 1 2 3 4 5
     b ← 3 2 1 4 1
     a sumsq b
10 8 10 32 26
```

[1]Dynamic functions.

Exercise 1.2 10 P.

Write a function `divs37` that, given an argument N, returns the number of natural numbers below N that are divisible by 3 or 7.

Exercise 1.3 10 P.

Write a function `sumsecond` that, given an even natural number N, returns the result of summing every second element in the argument vector (including the first element).

```
     a ← 1 2 3 4 5 7 92 2
     sumsecond a
101
```

Hint: Remember that the shape function `ρ` returns a vector; you can use indexing to get its content or you can use the function `⊃`, which picks the first element of a vector.

2 Minimum segment sum

Exercise 2.1 10 P.

Consider the following recursive implementation of an exclusive left-scan operator `xscanl`, which works with `http://tryapl.org`:

```
     xscanl ← { 0=⊃ρω: θ ◊ ωω , (αα ∇∇ (ωω αα ⊃ω)) 1↓ω }
```

Applying the operator to the list `1 2 3 4`, using the function `-` and with the right-neutral element 0, works as follows:

```
     (- xscanl 0) 1 2 3 4
0 ¯1 ¯3 ¯6
```

Notice that this result is different from the result of using APL's built-in inclusive "suffix scan":

```
     -\ 1 2 3 4
1 ¯1 2 ¯2
```

As demonstrated by Richard Bird, in his monograph "An Introduction to the Theory of Lists" (from 1986), the function $\odot$ = `{0⌊α+ω}` can be used to find the minimum segment sum, using the formula

```
     minsum ← {⌊/(⊙ xscanl 0) ω}
```

Demonstrate (by example) that the `minsum` function really works with the above `xscanl` operator.

Exercise 2.2 20 P.

Show that the function $\odot$ is not applicable in a parallel setting (i.e., with a parallel semantics of scan).

Hint: You need to demonstrate that $\odot$ lacks a particular important property.

3 Maximum element with index

Exercise 3.1 30 P.

Write a function `maxidx` that, given a sequence of integers returns the maximum element as well as its index (1-indexed).

Hint: For the best solution, you need to use nested arrays. You may choose to port your solution from the Futhark exercise.

4 Longest decrease

Exercise 4.1 10 P.

Write a function `longestdecr` that, given a sequence of numbers, returns the length of the longest decreasing sequence of numbers.

Hint: You probably need a few uses of the scan operator. This is a **DIFFICULT** problem. Remember to test your solution on a number of examples.