CPL Exercises
SCHEME II

Prof. Dr. Ulrike Pado

0: Getting Started

The Scheme interpreter available in LIDA is called `guile`. Use an editor to write your code and paste it into the interpreter (highlight the code in the editor, then middle-click into the interpreter).

1: Factorial with an Accumulator

We saw the naive implementation of the factorial function n! on the slides. Here is the code:

```
(define factorial
  (lambda (n)  (if (= n 0) 1
           (* n (factorial (- n 1))))))
```

Make a faster version acc-factorial with an accumulator that collects the results and can be returned once the base case is reached.

The wrapper for your function will be

```
(define fast-factorial
     (lambda (n) (acc-factorial n 1))
```

2: Reversing Lists

Use `fold-left` with an anonymous function to reverse a list argument. Given a list `'(1 2 3 4)`, the result should be `'(4 3 2 1)`.
Remember that `'()` denotes the empty list and that `cons` adds an atom to a list.

Note: In guile, `fold-left` is simply called `fold` and is available in a library that has to be loaded before use.

There are two ways of loading the srfi-1 library to get support for `fold`: Either start guile as `guile --use-srfi="1"` or start guile without this option and load the library manually as `(use-modules (srfi srfi-1))` once guile runs.

In either case, you will now be able to use `fold-right` and `fold` (which is fold-left).