Exercises for Week 3

3A. *(String expressions)*

Suppose that variable `s` contains the string 'ba', `t` contains 'na', and `u` contains 'banana'. Evaluate the following expressions:

(a) `s+t`

(b) `t+s`

(c) `2*s`

(d) `s*2`

(e) `s+(2*t) == u`

(f) `s in t`

(g) `t in u`

Suppose that variable `id` contains the string '01234567', `c` contains 120, and `g` contains 1260. Evaluate the following expression:

(h)
```
'Student %s has %d grade points and a GPA of %f.' \
% (id, c, g/c)
```

3B. *(String functions)*

Define functions to do the following:

(a) `bracket(s)` should return the string `s` enclosed in curly brackets. E.g., `bracket('xyz')` should return '{xyz}'.

(b) `trim_left(s)` should return the string `s` with all leftmost blanks removed. E.g., `trim_left('  xyz')` should return 'xyz'.

(c) `trim_right(s)` should return the string `s` with all rightmost blanks removed. E.g., `trim_right('xyz  ')` should return 'xyz'.

3C. *(Tuples)*

Suppose that variable `e` contains the tuple (7404, 'Jones', 6.50). Evaluate the following expressions:

(a) `e[0]`

(b) `e[2] * 40`

Execute the following statements:

(c) `pay = 1.2 * e[2]`

(d) `e[2] = pay`

3D. *(Lists)*

Suppose that variable `f` contains the list [1, 1, 2, 3, 5, 8, 13, 21]. Evaluate the following expressions:

(a) `f[4]`

(b) `f[-1]`

(c) `f[4:7]`

(d) `f[:2]`

(e) `[2,3,5] in f`

(f) `f + [f[-2]+f[-1]]`

3E. *(List functions)*

Define functions to do the following:

(a) `sum(ns)` should return the sum of the elements of the list `ns`, assuming that all elements are numbers. E.g., `sum([1,9,1,4])` should return 15, and `sum([])` should return 0.

(b) `concat(ss)` should return the concatenation of the elements of the list `ss`, assuming that all elements are strings. E.g., `concat(['ba', 'na', 'na'])` should return 'banana', and `concat([])` should return ''.

(c) `zip(xs, ys)` should return a list of pairs, in which each element of `xs` is paired with the corresponding element of `ys`. If one list is longer than the other list, the excess elements of the longer list should be discarded. E.g., `zip([1,2,3], [4,5,6])` should return [(1,4), (2,5), (3,6)], and `zip([1,2,3], [4,5,6,7])` should return the same.

(d) `tokens(s)` should break the string `s` down into a list of tokens. Here assume that a "token" is just a sequence of non-blank characters. E.g., `words('one fine day')` should return ['one', 'fine', 'day'].

3F. *(For-statements)*

Write for-statements to do the following:

(a) Compute and print the first `n` powers of 2 (i.e., 1, 2, 4, 8, 16, …).

(b) Compute and print the first `n` Fibonacci numbers (i.e., 1, 1, 2, 3, 5, 8, …, where each number is the sum of its two predecessors). Do *not* construct a list.

(c) Given a list of pairs, tabulate the list by printing all the values in two columns.

3G. *(List comprehensions)*

Suppose that variables `ms` and `ns` each contains a list of integers. Write list comprehensions to do the following:

(a) Yield a list of integers in which each integer is 1 greater than the corresponding integer in `ns`.

(b) Yield a list of booleans in which each boolean is True if the corresponding integer in `ns` is negative or False if non-negative.

(c) Yield a list containing only the positive integers in `ns`, in the same order.

(d) Yield a list of all possible pairs (`m`, `n`) in which `m` is taken from `ms` and `n` is taken from `ns`.