Computational Linguistics I, Winter 2006. Marcus Kracht

To be submitted: Friday, February 3, 2006.

[A 2.1] Let l be a list of strings, and x a single string. The function `zip` l x is defined as follows. If $l = [y_0; y_1; \ldots ; y_{n-1}]$ then `zip` l x is

$$y_0{}^\frown x^\frown y_1{}^\frown x \ldots {}^\frown x^\frown y_{n-1}$$

There are two ways to define this function (and I want you to try both): by recursion, and by using a single call to a function provided by the module `List`, that avoids recursion. (If you are curious what the code of the definition is, take a look at the source code.)

[A 2.2] Let x be a string of `0` and `1`. Such a string represents a number in the following way. If $x = c_{n-1}c_{n-2} \ldots c_1c_0$ then the number is

$$c_0 + 2c_1 + 2^2c_2 + \ldots + 2^{n-2}c_{n-2} + 2^{n-1}c_{n-1}$$

To facilitate its computation, I rewrite this as follows:

$$c_0 + 2(c_1 + 2(c_2 + \ldots + 2(c_{n-2} + 2c_{n-1}) \ldots))$$

Write a function that computes the translation from the binary string x to the number it represents.

[A 2.3] Write a module `SNum` that allows to manipulate sets of numbers.

[A 2.4] Write a function that tests whether a number is prime. There are at least two solutions (give me only one). One is to try to divide the number by all possible integers smaller than it and memorize whether one of the divisions works without remainder. The other is to use the type `SNum` and the functions of that module (that function call will eliminate the need to do iteration or recursion).