General Computer Science I (320101) Fall 2011
Assignment 7: SML Playground
(Given Nov.4., Due Nov. 10.)

10pt

Problem 7.1 (Apply substitutions)

Apply the substitutions $\sigma := [(h(a,b))/x], [(g(c,f(c),b))/y]$ and $\tau := [a/x], [b/c], [(h(b,c))/y]$ to the terms $s := h(g(x,y,c),x)$ and $t := g(y,h(y,x),a)$
(give the 4 result terms $\sigma(s)$, $\sigma(t)$, $\tau(s)$, and $\tau(t)$).

Prove of refute that substitution application is commutative , i.e. whether $[t_1/x^1_{\mathbb{A}}]([t_2/x^2_{\mathbb{A}}](s)) = [t_2/x^2_{\mathbb{A}}]([t_1/x^1_{\mathbb{A}}](s))$

30pt

Problem 7.2 (Inscriptions from Aiur)

While exploring an ancient Protoss temple, some Terrain ghosts found a script signed by Zeratul himself! In order to decrypt the hidden message of the script, Terrain scientists have figured out that the text (which appears on a single line), must be split on several lines such that there are no more than k letters per line. Also, the scientists know that all Protoss scripts contain "words" that are separated by spaces and are formed of letters, numbers and "," or ".". The string must be split at word boundaries (a word mus not be split on different lines). You are required to implement an SML function that will help the Terrains to decrypt the message, using the following signature:

```
val split = fn : string * int -> string list
```

Example:

```
- split("Attack the zergs on Tarsonis.", 10);
val it = ["Attack the", "zergs on", "Tarsonis."] : string list
```

Hint: Consider using the SML functions `explode` and `implode`. Also, note the required output format (no spaces at the beginning or end of the strings).

20pt

Problem 7.3 (Mutual recursion in SML)

Implement functions for the following recursive/ mutually recursive functions:

- the Hofstadter H sequence:
$$H(n) = \begin{cases} 0, & \text{if } n = 0 \\ n - H(H(H(n-1))), & \text{if } n > 0 \end{cases}$$
- the Hofstadter Q sequence:
$$Q(n) = \begin{cases} 1, & \text{if } n = 1 \text{ or } 2 \\ Q(n - Q(n-1)) + Q(n - Q(n-2)), & \text{if } n > 2 \end{cases}$$
- the Hofstadter male and female sequences:
$$male(n) = \begin{cases} 0, & \text{if } n = 0 \\ n - female(male(n-1)), & \text{if } n > 0 \end{cases}$$
$$female(n) = \begin{cases} 1, & \text{if } n = 0 \\ n - female(male(n-1)), & \text{if } n > 0 \end{cases}$$

- the following mutually recursive functions:

$$a(n) = \begin{cases} 1, & \text{if } n = 0 \\ 2, & \text{if } n = 1 \\ a(n-2) * Q(n+1), & \text{if } n > 1 \end{cases}$$

$$b(n) = \begin{cases} c(1), & \text{if } n = 0 \\ c(2), & \text{if } n = 1 \\ c(n-2), & \text{if } n > 1 \end{cases}$$

$$c(n) = \begin{cases} 1, & \text{if } n = 0 \\ a(n-1) * c(n-1), & \text{if } n > 0 \end{cases}$$

$$d(n) = \begin{cases} 0, & \text{if } n = 0 \\ b(a(n)), & \text{if } n > 0 \end{cases}$$

Please raise appropriate exceptions when the arguments of the functions are negative.

30pt

Problem 7.4 (Permutations)

For a set $A := \{1, 2, \ldots, n\}$, a permutation σ is a bijective function $\sigma \colon A \to A$. Permutation multiplication can be defined as:

$$(\sigma \cdot \pi)(i) := \sigma(\pi(i))$$

Intuitively, we can also define the "raise to power"operation for permutations as multiplication applied on itself:

$$\sigma^k := \underbrace{\sigma \cdot \sigma \cdot \ldots \cdot \sigma}_{k \text{ times}}$$

We can also define the "identity permutation" as: $id(i) := i$. One of its important properties is that $id \cdot \sigma = \sigma \cdot id = \sigma$.

Your task is to write an SML function `findPower` that, given a permutation σ, will return the minimum power $k > 0$ for which $\sigma^k = id$. The function should have the following signature:

```
val findPower = fn : int list -> int
```

Examples:

```
- findPower [1,2,3];
val it = 1 : int
- findPower [3,1,2];
val it = 3 : int
- findPower [5,1,9,12,3,4,8,7,2,6,11,10];
val it = 20 : int
```

Note: Your solution should also check that the input permutation is valid and raise an exception if it is not.

Hint: You might consider writing a function for product between two permutations, and also a function for raising a permutation to a power.