Grammaires ; récupération de données

Info de dernière minute : On trouvera sur `http://www.banane.be/textes.php` (rubrique : "à méditer") l'annonce d'une découverte en informatique théorique tout à fait spectaculaire. Elle est bien trop récente pour tomber à l'écrit dès cette année, mais ceux qui présentent les ENS auront à mon avis tout intérêt à aller voir ce qu'il en est avant les oraux...

1 Sur les grammaires

Exercice 1 Donner une grammaire algébrique pour :
- $\{a^n b^n \mid n \in \mathbb{N}\}$,
- L'ensemble des mots bien parenthésés (alphabet : $\{(,)\}$,puis $\{a, b, (,)\}$)
- Les mots qui possèdent le même nombre de a que de b.
- Les mots qui possèdent deux fois plus de a que de b.
- (hard) $\{a^{2^n} \mid n \in \mathbb{N}\}$.

Exercice 2 Pour chacun des langages de l'exercice précédent, donner un automate à pile associé. *A vous de voir une définition raisonnable d'un automate à pile !*

Exercice 3 Montrer que $S \to SbS \mid abS \mid a$ est ambigüe. La "désambigüiser".

Exercice 4 Quel est le langage généré par la grammaire suivante : $S \to aS \mid aSbS \mid \varepsilon$?

2 De la vraie informatique

Toute ressemblance avec un logiciel tout pourri existant ou ayant existé serait bien entendu fortuite

On dispose en entrée d'un fichier de plusieurs mégaoctets nommé `toto.slx` d'un format mystérieux. Il contient une liste d'élèves, avec une note associée à chacun.
L'objectif est de reconstituer un fichier de quelques centaines d'octets contenant les mêmes informations, mais sans toutes les saletés contenues dans le fichier initial. C'est une fonction `slx2txt` qui sera chargée de faire ce travail. Elle sera du type `string -> unit` (on donne le nom du fichier, sans l'extension).

1. Charger le programme `porky.ml` ; lire le code de `create_porky_file` pour voir ce qu'il fait. L'exécuter et regarder le fichier produit (ça doit vous rappeller quelque chose si vous avez déjà ouvert des fichiers G(Ré]çà"è*$ avec un éditeur...).

2. Pour se mettre en route et faire une première manipulation de fichiers, écrire une fonction `taille` de type `string -> int` qui calcule la taille d'un fichier (en nombre d'octets). Typiquement, on fait des lectures d'octets jusqu'à ce que le système lève une exception `End_of_file`, que l'on peut récupérer grâce au schéma suivant :

```
try
  .....
with End_of_file -> !c;;
```

3. Le programme `slx2txt` devra ouvrir un fichier en lecture, récupérer le nombre d'élèves, puis créer une liste (ou un tableau) contenant ces élèves (couple nom-note), et enfin réécrire le tout dans un fichier `toto.txt` dans un format lisible.

   Dans un premier temps, écrire une fonction `get_number` prenant en entrée un canal de lecture (donné par le programme principal) et retournant le nombre d'élèves détecté.

   On pourra noter que `seek_in file ((pos_in file)-1)` *permet de revenir un cran en arrière dans un fichier...*

   A ce niveau, votre programme principal peut avoir la tête :

```
let slx2txt name=
  let in_file=open_in (name^".slx") and out_file=open_out (name^".txt")
  in
    print_int (taille (name^".slx"));
    print_int (get_number in_file);
    close_in in_file; close_out out_file;;
```

   et peut être testé avec : `slx2txt (make_porky_file 50)`.

4. Ecrire une fonction `next_student` de type `in_channel -> string * int` qui retourne le nom et la note du premier étudiant rencontré dans le canal fourni (depuis la position courante de lecture).
5. Ecrire les données récupérées dans un fichier `.txt` lisible par un humain normalement constitué.
6. Pour finasser : modifiez votre programme pour que les noms apparaissent dans l'ordre lexicographique croissant (on pourra utiliser le module `sort`).
7. Recommencer en vrai Caml chicho-fluxien.