Exercice 1

Introduction

Vous trouverez sur la page Web du cours une version de l'interpréteur PROLOG vu au cours, et que nous allons utiliser pour cette série. Nous nous contenterons ici de l'utiliser, sans le modifier d'aucune manière.

Notez encore que l'interpréteur PROLOG vous permet de charger le contenu d'un fichier au moyen de la commande `:l` semblable à celle de `scalaint`. Pour quitter, utilisez `:q`.

Exercice

Le but de cet exercice est de réaliser un programme PROLOG permettant de jouer avec les « cadavres exquis ». Les cadavres exquis sont des phrases obtenues en combinant des mots aléatoires en s'assurant que la phrase produite soit grammaticalement correcte.[1] En fait, ce programme produira *toutes* les phrases possibles à partir d'un vocabulaire donné, mais cela n'est pas important.

La réalisation d'un tel programme est fort simple : il suffit de décrire la structure grammaticale d'une phrase, et de donner des règles pour en produire les différentes composantes.

Pour commencer avec un cas très simple, admettons qu'une phrase est composée d'un simple sujet suivi d'un verbe intransitif. Le programme PROLOG suivant permet de produire toutes les phrases de cette forme avec les sujets *Jean* et *Marie*, et les verbes *mange* et *dort*.

```
sujet(jean).
sujet(marie).
verbe(mange).
verbe(dort).
phrase([S,V]) :- sujet(S), verbe(V).
```

Notez que ce programme permet aussi bien de produire toutes les phrases de cette forme, avec le vocabulaire donné, que de vérifier si un sujet et un verbe donnés forment bien une phrase.

[1] Le nom *cadavre exquis* vient semble-t-il du premier poème ainsi réalisé : « Le cadavre exquis boira du vin nouveau »

Partie 1 Sur la base de cette idée, commencez par définir une structure plus complète pour les phrases. On peut par exemple imaginer qu'une phrase est composée d'un sujet, d'un verbe transitif et d'un complément. Le complément est à son tour composé d'un article, d'un adjectif et d'un nom. N'oubliez pas de définir un certain nombre d'articles, d'adjectifs et de noms.

Partie 2 Un problème que vous constaterez immédiatement est celui du genre : si *le* et *la* sont des articles, *belle* et *grand* des adjectifs et *table* et *vélo* des noms, le programme accepte *la belle vélo* comme complément, ce qui est grammaticalement faux. Modifiez donc votre programme afin qu'il n'admette comme compléments que ceux composés d'un article, d'un adjectif et d'un nom de même genre.

Partie 3 Finalement, pour corser encore un peu le tout, faites en sorte que votre complément puisse comporter zéro, un ou deux adjectifs avant le nom. Bien entendu, tous ces adjectifs devront avoir le même genre que le nom qu'ils qualifient. Pour ce faire, utilisez les listes vues au cours, définies au moyen des constructeurs `nil` et `cons`.

Exercice 2

En utilisant l'interpréteur de la semaine précédente et les notes du cours de la semaine 5, écrivez des versions LISP des fonctions Scala suivantes :

LISP	SCALA
`map xs f`	`xs.map(f)`
`flatMap xs f`	`xs.flatMap(f)`
`foldLeft xs f e`	`xs.foldLeft(e)(f)`
`foldRight xs f e`	`xs.foldRight(e)(f)`
`reverse xs`	`reverse(xs)` (en utilisant foldLeft)

Définissez les fonctions comme fonctions globales, c'est-à-dire `(def (fn args) body)`.