Lambda calcul et programmation fonctionnelle

mini-projet : générateur de texte – travail à rendre

Le but de ce projet est d'écrire un programme qui prendra en entrée un texte (qu'on appellera texte d'apprentissage) et qui générera en sortie un nouveau texte à partir de ce qui aura été appris.

La phase d'apprentissage, qui génère une table de correspondance de type `[(String, [String])]` a déjà été traitée en partie 1.

Pour la partie de génération, nous procéderons comme suit.

Étape 1 : prendre le premier mot du texte d'apprentissage, et l'ajouter à la liste des mots à afficher.

Étape 2 : dans la table de correspondance, prendre au hasard un des mots qui peut suivre le mot qui vient d'être ajouté, et l'ajouter à son tour. S'il n'y a aucun mot qui peut suivre ce mot, alors on a terminé : aller à l'étape 3. Sinon, on répète l'étape 2, afin d'ajouter encore un mot.

Étape 3 : afficher le résultat.

Exemple :

Le texte d'apprentissage est « Je suis en train de découvrir le langage de programmation Haskell, et je suis très content. Je pense que Haskell est le meilleur langage de programmation au monde. »

La table de correspondance qui sera générée est :

```
[("je", ["suis", "suis", "pense"]),
 ("suis", ["en", "très"]),
 ("en", ["train"]),
 ("train", ["de"]),
 ("de", ["découvrir", "programmation", "programmation"]),
 ("découvrir, ["le"]),
 ("le, ["langage", "meilleur"]),
 ("langage, ["de", "de"]),
 ("programmation", ["haskell", "au"]),
 ("haskell", ["et", "est"]),
 ("et", ["je"]),
 ("très", ["content"]),
 ("content", ["je"]),
 ("pense", ["que"]),
 ("que", ["haskell"]),
 ("est", ["le"]),
 ("meilleur", ["langage"]),
 ("au", ["monde"]),
 ("monde", [])]
```

(voir partie 1 de ce projet).

À partir de là, nous prendrons le premier mot du texte : « je ». Les mots qui peuvent suivre « je » sont « suis », « suis » et « pense ». Nous prendrons par exemple « suis ». Les mots qui peuvent suivre « suis » sont « très » et « en ». Nous prendrons par exemple « en ». Et ainsi de suite.

Au final, nous pourrons avoir la liste suivante : [« je », « suis », « en », « train », « de », « découvrir », « le », « langage », « de », « programmation », « au », « monde »]. L'algorithme s'arrête là parce que « monde » n'a pas de successeur.

Nous afficherons donc « je suis en train de découvrir le langage de programmation au monde » (ce qui ne veut pas dire grand-chose mais peu importe.

Vous aurez besoin d'écrire plusieurs fonctions pour résoudre ce problème. L'une des fonctions que vous devrez écrire est :

```
motAuHasard :: [String] -> [Int] -> String
motAuHasard listeDeMots listeDeNombresAléatoires = … ?
```

qui prend au hasard un mot dans une liste donnée en paramètre. Il faut pour cela utiliser une liste de nombres aléatoires, comme nous l'avons vu en cours. Si la liste de mots est `["suis", "suis", "pense"]`, on choisir un nombre i au hasard entre 0 et 2 (puisqu'il y a trois éléments) et on renverra le ième élément de la liste.

Le travail est à envoyer à votre enseignant de TP et à effectuer en binôme ou seul (au choix).