Interrogation écrite 3

INF 201 — IMA4— 30/03/2023 — 30 minutes

Exercice 1. (/4) La suite de Padovan est définie par:

$$P_0 = P_1 = P_2 = 1 \quad \text{et} \quad P_{n+3} = P_{n+1} + P_n$$

Donner la spécification de `padovan(n)`:

PROFIL:

SÉMANTIQUE:

EXEMPLE: `padovan(4)` =

Donner une implémentation **récursive** en OCaml de `padovan(n)`:

Exercice 2. (/4) On étudie une fonction mystère définie par:

```
let rec mystere a b = if b = 0 then a else mystere (a+1) (b-1);;
```

Donner le profil de `mystere`:

PROFIL:

Quel sera le résultat de `mystere 99 2` ?

Et de `mystere 333 51` ?

Et enfin de `mystere (-2) (-3)` ? (petit piège !)

Exercice 3. (/14) On souhaite écrire dans cet exercice un petit correcteur orthographique: c'est à dire un programme qui prend en entrée un texte et un dictionnaire, puis pour chaque mot du texte vérifie qu'il se trouve dans le dictionnaire, si c'est le cas le mot n'est pas changé, sinon il est remplacé par le mot le plus proche. On représentera un texte par une **liste de mots** qui seront représentés eux-mêmes représentés par des **listes de caractères.** Un dictionnaire sera également représenté par un type texte mais **non vide**, c'est-à-dire contenant au moins un mot.

```
let maison:mot = ['m';'a';'i';'s';'o';'n'];;
let la:mot = ['l';'a'];;
let t:texte = [la;maison];;
```

Question 1. (/1) Définir des types pour représenter un mot et un texte.

Question 2. (/1) On suppose que l'on dispose d'une fonction `max` qui renvoie le maximum entre deux entiers et d'une fonction `min` qui renvoie le minimum entre deux entiers. En déduire une fonction `min3` qui renvoie le minimum entre trois entiers.

Question 3. (/2) Écrire une fonction `longueur_mot` qui renvoie la longueur d'un `mot`. On pose `longueur_mot [] = 0`.

Question 4. (/1) Rappeler quelle fonction du module `List` permet d'accéder au premier élément d'une liste. Quelle est la restriction importante sur cette fonction ?

On rappelle que la fonction qui permet d'accéder à une liste privée de son premier élément est `List.tl`.

Pour calculer la ressemblance de deux mots a et b on va utiliser la distance de Levenshtein qui est définie récursivement par:

$$\text{lev}(a,b) = \begin{cases} \max(|a|,|b|) & \text{si} \quad \min(|a|,|b|) = 0 \\ \text{lev}(a-1,b-1) & \text{si} \quad a[0] = b[0] \\ 1 + \min \begin{cases} \text{lev}(a-1,b) \\ \text{lev}(a,b-1) \\ \text{lev}(a-1,b-1) \end{cases} & \text{sinon} \end{cases}$$

Où $|a|$ représente le nombre de caractères dans a, $a-1$ représente le mot a privé de son premier caractère et $a[0]$ le premier caractère de a. Même chose pour b. Par exemple:

$$\text{lev}(''\text{maison}'',''\text{maison}'') = 0$$

$$\text{lev}(''\text{question}'',''\text{kestion}'') = 2$$

Question 5. (/3) Écrire une fonction `lev a b` qui calcule la distance de Levenshtein entre deux mots. On utilisera des fonctions écrites précédemment.

Question 6. (/3) Écrire une fonction `mot_lpp mot dictionnaire` qui compare un mot à tous les mots d'un dictionnaire et renvoie le mot le plus proche (qui peut être le même si le mot est bien orthographié!). On rappelle qu'un dictionnaire n'est jamais vide et que le mot retourné doit être dans le dictionnaire.

Question 7. (/3) Écrire une fonction `correcteur texte dictionnaire` qui corrige les mots dans un texte à partir d'un dictionnaire. On appliquera la fonction `mot_lpp` sur chaque mot du texte.