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: `int -> int`

SÉMANTIQUE: Calcule le nième terme de la suite de Padovan.

EXEMPLE: `padovan(4)` = 2

Quelques confusions entre profil et sémantique, attention ça fait quand même 3 mois qu'on en parle … Confusion aussi: `padovan(n)` calcule bien le nième terme, pas le n+3-ième.

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

Avec le match, solution plus élégante, attention à ne pas se tromper, on additionne les termes aux rangs $n-2$ et $n-3$!

```
let rec p n = match n with
  |0|1|2->1
  |_->(p (n-2)) + (p (n-3));;
```

Également possible avec un `if...then...else`:

```
let rec p n = if (n=0||n=1||n=2) then 1
              else (p (n-2)) + (p (n-3));;
```

Taux de réussite: 5/26. Encore beaucoup d'erreurs à déplorer dans le match, incompréhension sur la suite elle-même.

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: `int->int->int`

Quel sera le résultat de `mystere 99 2` ?

101

Et de `mystere 333 51` ?

384

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

ne termine pas.

Taux de réussite: 16/26, très bien réussi, presque tout le monde a compris qu'il s'agissait de l'addition sur les entiers posisitfs.

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.

```
type mot = char list;;
type texte = mot list;;
```

Taux de réussite: 14/26. Pas questiion d'utiliser des `string` ici, il est bien dit dans l'énoncé de traiter des listes de caractères !

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.

```
let min3 a b c = min a (min b c);;
```

Taux de réussite: 15/26. Beaucoup de solutions inutilement complexes à base de max et de conditions. Cette fonction avait été vue en TP !

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

```
let rec longueur_mot l = match l with
  |[] -> 0
  |t::q -> 1 + longueur_mot q;;
```

Taux de réussite: 18/26. Un classique vu en TD, très bien réussi. Je suis content.

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 ?

`List.hd` et non pas `hd.List`, `List.head`, `List.fst`, ...

La liste doit être non vide !

Taux de réussite: 8/26 Décevant pour une question vue à plusieurs reprises en TP/TD.

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.

```
let rec lev a b =
  let ma = longueur_mot a and mb = longueur_mot b in
  if min ma mb = 0 then max ma mb
  else if List.hd a = List.hd b then lev (List.tl a) (List.tl b)
  else 1 + (min3 (lev (List.tl a ) b) (lev a (List.tl b)) (lev (List.tl a) (List.tl b)))
;;
```

Question bien plus difficile que les précédentes et qui a causé pas mal de soucis. Bien sûr en `OCaml`, `a-1` ou `|a|` n'a PAS de sens ! Il faut utiliser les fonctions implémentées avant ! Taux de réussite: 5/26

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.

```
let rec mot_lpp (mot:mot) (dictio:texte) = match dictio with
  |[] -> failwith "Dictionnaire vide."
  |[t] -> t
  |t::q -> let motq = mot_lpp mot q in
           if (lev mot t) < (lev mot motq) then t else motq;;
```

Ici aussi, question peu abordée, le cas terminal était un peu particulier, il faut bien sûr utiliser `lev`. Taux de réussite: 2/26

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.

```
let rec correcteur (texte:texte ) (dictio:texte) = match texte with
  |[] -> []
  |t::q -> (mot_lpp t dictio)::(correcteur q dictio);;
```

Question plus facile que les deux précédentes, il fallait lire jusqu'au bout ! Il s'aggit d'appliquer une fonction à tous les éléments d'une liste. Un classique. Taux de réussite: 5/26.

Dans l'ensemble, c'est bien mieux que la dernière fois, la récursivité et les listes semblent être des notions acquises par la majorité d'entre vous. Attention cependant à ceux qui ont eu des difficultés sur les exercices 1 et 2 ou les premières questions de l'exercice 3…

$\bar{x}$	10.83
σ_x	5.63
$x_{\max}$	20
$x_{\min}$	0