Langages fonctionnels (INFO406)
Examen – Session 1

L2 informatique, CMI Informatique, Maths Info, Peip option info
Mercredi 22 mars 2023 – 13 h 15 à 15 h 15 (+ 1/3 temps)

Remarques :

- *Le sujet comporte 4 pages.*
- *Les exercices sont indépendants : vous les traiterez dans l'ordre que vous souhaitez.*
- *Le barème est donné à titre indicatif.*

Consignes :

- *Précisez systématiquement le **profil** des fonctions que vous avez à écrire dans ce qui suit.*
- *Les fonctions récursives que vous écrivez doivent engendrer des **processus récursifs.***
- *Vous introduirez toute fonction jugée utile à la définition des fonctions qui vous sont demandées. Dans le cadre d'une **analyse fonctionnelle descendante**, vous donnerez les profils des fonctions introduites, vous indiquerez ce qu'elles font et les utiliserez comme si elles existaient. Vous les définirez ensuite.*
- *Des fonctions de base pour le traitement des chaînes de caractères vous sont fournies (cf. ci-dessous). **Pensez à les utiliser.***
- *N'hésitez pas à passer une question si vous ne parvenez pas à écrire la fonction qui y est demandée. Vous pourrez malgré cela utiliser cette fonction dans la suite de votre devoir.*

Fonctions de base pour le traitement des chaînes de caractères.

```
open String;;

let est_vide = function ch -> (ch = "");;
(** string -> bool
teste si son paramètre est une chaîne vide *)

let prem_lettre = function ch -> sub ch 0 1;;
(** string -> string
renvoie la première lettre d'une chaîne de caractères *)

let reste = function ch -> sub ch 1 ((length ch) - 1);;
(** string -> string
renvoie la chaîne de caractères passée en paramètre privée de sa première lettre *)
```

Rappel : La fonction ***int_of_string*** (resp. ***float_of_string***) traduit une chaîne de caractères représentant un entier (resp. un flottant) en le nombre correspondant ; la fonction ***string_of_int*** (***string_of_float***) fait le contraire.

Question 1 – Devine qui je suis (4 pts)

1.1 - Quel est le profil de ***mystere,*** définie ci-après ? Justifier votre réponse. Vous pourrez pour cela avoir recours aux équations aux types (Attention : pas de point si réponse non justifiée).
1.2 - Donnez deux exemples d'application de la fonction ***mystere***, avec le résultat obtenu.
1.3 - Que fait la fonction ***mystere*** ?
1.4 - Quel type de processus (récursif ou itératif) engendre-t-elle ? Justifiez votre réponse.

```
let rec mystere = function a ->
  if (length a) <= 1
    then 0
  else
    let b = (prem_lettre a) and c = (reste a)
    in
      let d = (prem_lettre c)
      in
        if b = d
          then 1 + mystere (reste c)
        else mystere c;;
```

Question 2 – Où l'on calcule les résultats du bac... (13 pts : 2 – 1 – 2 – 2 – 3 – 3)

Les épreuves préliminaires du bac ont lieu en ce moment. On voit fleurir un peu partout sur internet des simulateurs de note qui permettent de calculer le résultat au fur et à mesure de la saisie des notes. Vous allez ici écrire quelques unes des fonctions qui pourraient faire partie de ces simulateurs.

On suppose que l'on dispose, pour chaque type de bac, de la liste des matières à passer (obligatoires et options) avec leurs coefficients. Ces données sont enregistrées dans des chaînes de caractères avec en alternance le nom d'une matière suivi du coefficient associé. Les informations sont séparées par un espace de sorte qu'il est aisé de les repérer. Nous appellerons de telles données des ***tMatieresBac.***

Exemple : "FrançaisEcrit 5 FrançaisOral 5 Philosophie 4 HistGéo1 3 HistGéoT 3 SpéArts 16 .. "

Les résultats d'un élève sont également enregistrés dans une chaîne de caractères qui a la même structure : une alternance de noms de matière et de notes obtenues dans la matière. Ces résultats sont personnalisés et construits en fonction du type de bac dans lequel l'élève est inscrit ainsi qu'en fonction de ses choix d'options. Les notes sont initialisées à -1. Elles sont remplacées par celles effectivement obtenues au fur et à mesure de la correction des épreuves et de leur saisie. Les noms de matières correspondent forcément à des noms contenus dans un ***tMatieresBac.*** Ils n'apparaissent en revanche pas le même ordre. Nous appellerons de telles données des ***tResBac***.

Exemples :

- une liste initiale : "HistGéo1 -1 Maths -1 Physique -1 NSI -1 FrançaisEcrit -1"
- une liste en partie complétée : "HistGéo1 12 Maths -1 Physique 7.5 NSI 14.3 Français 10" L'élève attend les résultats de Maths.

Vous allez manipuler ces structures pour écrire les fonctions demandées ci-après. Vous supposerez l'existence des fonctions suivantes, que vous utiliserez sans les définir :

- **get_mot : string → string**
  Prend en paramètre une chaîne de caractères *ch* contenant des mots séparés par des espaces et renvoie le premier mot de *ch*.
- **get_reste : string → string**
  Prend en paramètre une chaîne de caractères contenant des mots séparés par des espaces et renvoie la chaîne privée de son premier mot.

2.1 – (2 pts) – Soit une chaîne de caractères permettant de représenter une structure dans laquelle des ***clés*** sont associées à des ***valeurs***. De telles chaînes ont la forme suivante :

"cle1 valeur1 cle2 valeur2 cle3 valeur3 …"

Elles permettent de représenter des objets définis par leurs propriétés (les clés), chacune ayant une valeur : la ***clé-i*** a pour valeur ***valeur-i***. Les *tResBac* et *tMatièresBac* sont des exemples de tels objets.

Utilisez les fonctions *get_mot* et *get_reste* pour définir la fonction ***get_val*** qui prend en paramètres :

- une chaîne de caractères de la forme "cle1 valeur1 cle2 valeur2 cle3 valeur3 …",
- une chaîne de caractères correspondant à une clé,

et renvoie la valeur associée à la clé.

2.2 – (1 pt) – Utilisez la fonction ***get_val*** pour définir les fonctions permettant de récupérer les données de base contenues dans des *tResBac* et des *tMatièresBac* **:**

- **get_coef : string → string → int**
  Prend en paramètres une chaîne de la forme *tMatièresBac* **,** le nom d'une matière et renvoie le coefficient associé. C'est 4 pour la philosophie dans l'exemple donné en page 2.
- **get_note : string → string → float**
  Prend en paramètres une chaîne de la forme *tResBac***,** le nom d'une matière et renvoie la note obtenue dans la matière. C'est 14.3 pour NSI dans l'exemple donné en page 2.

2.3 – (2 pts) – Ecrire la fonction ***coefs_eleve*** qui renvoie une chaîne de caractères contenant les coefficients des matières à passer au bac pour un élève donné. Cette fonction prend en paramètres :

- les résultats au bac obtenus par un élève (un *tResBac***)**,
- la liste des matières avec leurs coefficients (un *tMatièresBac*)

Le résultat de la fonction ne contiendra que les coefficients des matières, pas leurs noms. Ils seront enregistrés dans le même ordre que celui des notes dans le *tResBac.*

Exemple :

- résultat obtenu par un élève = "HistGéo1 12 Maths -1 Physique 8 FrançaisEcrit -1"
- les coefficients du bac = "FrançaisEcrit 5 HistGéo1 3 Maths 6 Physique 3 NSI 6 "
- ⇒ le résultat de la fonction = "3 6 3 5"

2.4 – (2 pts) – Ecrire la fonction ***total_points*** qui renvoie le nombre de points d'ores et déjà acquis par l'élève au moment où il lance le calcul. Cette fonction prend en paramètres les résultats obtenus par un élève (un **tResBac)**, la liste des coefficients correspondant (même format que le résultat de coefs_eleve : une chaîne contenant des entiers, séparés par des espaces) - et renvoie un flottant.

Le nombre de points est le produit cartésien entre la liste des notes et celle des coefficients : somme des produits (note de matière-i * coef de matière-i). Le calcul se fera en utilisant la valeur 0 pour les notes manquant. Pour rappel, ce sont celles qui sont à -1 dans la chaîne des résultats **tResBac**.

2.5 – (3 pts) – Nous allons maintenant définir les fonctions qui sont sensées avoir le plus de succès dans les simulateurs de calcul des notes : celles calculant les moyennes. Deux calculs différents sont pour le moment envisagés :

- le calcul de la moyenne effective actuelle (**mea**)
- le calcul de moyenne théorique (**mth**)

Si les 2 prennent en compte le nombre de points acquis, ils diffèrent par les coefficients considérés :

- la moyenne effective actuelle considère la totalité des coefficients pour faire un calcul de moyenne, y compris ceux des matières pour lesquelles les notes ne sont pas connues;
- la moyenne théorique neutralise les coefficients pour lesquels les notes ne sont pas connues. Cela signifie que le coefficient n'est pas compté.

Exemple :

- résultat obtenu par un élève = "HistGéo1 12 Maths -1 Physique 8"
- les coefficients du bac = "FrançaisEcrit 5 HistGéo1 3 Maths 6 Physique 3 NSI 6 "
- les coefficients des matières du bac de l'élève = "3 6 3"
- total des points = (12 * 3) + (0 * 6) + (8 * 3) soit 60
- calcul **mea** = total points / (3 + 6 + 3) = 60 / 12 = 5
- calcul **mth** = total points / (3 + 0 + 3) = 60 / 6 = 10
  La note de maths manquant, le coefficient correspondant a été neutralisé

Le mea fournit un calcul exact étant donné les notes connues. Le mth permet de se projeter dans l'avenir en supposant que l'élève obtiendra des notes équivalentes à celles qu'il a déjà obtenues.

Définir la fonction générique ***calcul_moyenne*** qui permettra de réaliser l'un ou l'autre des calculs de moyennes mea et mth. Vous préciserez son profil. Vous définirez ensuite les fonctions ***calcul_mea*** et ***calcul_mth*** par application partielle de ***calcul_moyenne***.

2.6 – (3 pts) – Les élèves aiment beaucoup à calculer, lorsqu'ils ont des résultats partiels, les notes qu'il leur faut avoir dans les matières pour lesquelles la note n'est pas encore connue, pour obtenir la moyenne et, ainsi, réussir. Définir la fonction ***quelle_note_faut_il*** qui réalise cette fonctionnalité. Elle renverra la note à obtenir pour avoir 10 de moyenne et obtenir ainsi son bac. On considérera que c'est la même note qui est attribuée à toutes les matières pour lesquelles la note est manquante.

Vous définirez le profil de cette fonction et procéderez à une analyse par décomposition fonctionnelle, en introduisant toute fonction que vous jugerez utile à sa définition. Il vous faudra définir les fonctions introduites.

Question 3 – Où l'on joue avec les mots (6 pts)

On cherche à déterminer des sous-séquences de caractères communes à des chaînes de caractères données. Soit **C** une chaîne de caractères notée C_1, C_2, C_3, …, C_N où les Ci représentent chacun un caractère. Une sous-séquence de **C** est une chaîne de caractères Ci1, Ci2, …, Cik, les Cij étant des caractères de **C** pris dans leur ordre d'apparition dans **C**.

Par exemple, si **C** = "bateau" alors "ba", "bat", "bt" ou encore "aau" sont des sous-séquences de **C** (il y en a d'autres, bien sûr).

Si l'on considère deux chaînes de caractères ch1 et ch2, les sous-séquences communes à ch1 et ch2 sont les sous-séquences de ch1 qui sont aussi sous-séquences de ch2.

Par exemple :

ch1	ch2	Des exemples de sous-séquences communes (liste non exhaustive)
"bateau"	"banane"	"ba", "bae", "baa", "aa", "ae"
"bateau"	"abattue"	"batu", "bate", "bau", "atu", "au", ...
"ananas"	"banane"	"anan", "aa", "aan", ...
"ananas"	"maracas"	"aaas" , "aaa" , "as" , ...

3.1 – Ecrire la fonction ***verif_sseq_commune*** qui vérifie qu'une chaîne de caractères donnée est une sous-séquence commune à deux chaînes données.

3.2 – Ecrire la fonction ***plus_longue_sseq_commune*** qui calcule la taille de la plus longue sous-séquence commune à deux chaînes de caractères données.
A noter : ce n'est pas (du tout !!) une bonne idée que de chercher à construire les sous-séquences.