− . − −./... − −/. − .. .. − ./

1 Introduction

Rappel : il est strictement INTERDIT d'utiliser l'opérateur @.

1.1 L'alphabet Morse

"Inventé en 1835 pour la télégraphie, ce codage de caractères assigne à chaque lettre, chiffre, et signe de ponctuation une combinaison unique de signaux intermittents. Considéré comme le précurseur des communications numériques, le code morse a depuis le 1^{e} février 1999 été délaissé au profit d'un système satellitaire pour les communications maritimes. Il est cependant encore utilisé à l'heure actuelle par les militaires pour la transmission en aviation par les systèmes de balises ; il est également pratiqué par des amateurs comme de nombreux scouts et radioamateurs."
Source : Wikipédia.

Le morse nous donne un format binaire des lettres et ponctuations que l'ont connaît. Une lettre n'est composée que de '.' (impulsion courte) et de '-' (impulsion longue) et la conversion a été faite "intelligemment" : Les lettres les plus utilisées sont les plus courtes et les moins compliquées (exemple : le 'e' devient '.').Le but de ce TP est de convertir et manipuler l'alphabet morse. Une solution pourrait être de considérer les lettres morses sous forme de chaînes de caractères. Pour rendre la chose plus intéressante, et de facto plus manipulable et plus $< fun >$, nous allons considérer qu'une lettre est une liste de caractères '.' ou '-', et qu'un mot est une liste de lettres, donc une liste de liste de caractères.

Exemples :

```
#latin_letter_to_morse 'X';;
 - : char list = ['-'; '.'; '.'; '-']

#latin_word_to_morse "42";;
 - : char list list = [['.'; '.'; '.'; '.'; '-']; ['.'; '.'; '-'; '-'; '-']]
```

2 Level 0 : Les Must-do

2.1 Egalité

Écrire la fonction *are_equal* qui teste si deux listes sont égales.

```
#are_equal [1, 2, 3] [1, 2, 3];;
 - : bool = true
#are_equal ['a', 'b', 'c'] ['a', 'b', 'c', 'd'];;
 - : bool = false
```

2.2 Concaténation

Écrire la fonction *append* qui à partir de deux listes colle la deuxième à la fin de la première.

```
#append [1; 2; 3] [42; 0; 21];;
 - : int list = [1; 2; 3; 42; 0; 21]
```

2.3 Inversion

Écrire la fonction *reverse* qui inverse une liste.

```
#reverse ['a'; 'l'; 'l'; 'o'];;
 - : char list = ['o'; 'l'; 'l'; 'a']
```

2.4 Impression

Écrire une fonction qui affiche une liste de caractères.

3 Level 1 : Les lettres

3.1 Validité

Écrire une fonction qui renvoie si une liste de caractère est bien en morse.

```
#is_morse ['-', '-', '-'];;
 - : bool = true
#is_morse ['.', '.', 'k'];;
 - : bool = false
```

3.2 Conversion

Écrire une fonction qui renvoie la version morse de la lettre passée en paramètre

```
#letter_to_morse 'S';;
 - : char list = ['.'; '.'; '.']
```

4 Level 2 : Les mots

4.1 Conversion

Écrire la fonction qui convertit une chaîne de caractères latins en liste de lettres morses telles qu'on les a écrites précédemment.

```
#word_to_morse ['S', 'O', 'S'];;
 - : char list list = [['.'; '.'; '.']; ['-'; '-'; '-']; ['.'; '.'; '.']]
```

4.2 Une autre façon de voir les choses

Une des écritures standard du morse est de séparer les lettres par des espaces ' '. Ici, nous allons nous autoriser l'utilisation de ce caractère pour convertir une liste de liste de caractères (un mot en morse) par une liste de caractère.

```
#to_single_list  [['-'; '-'; '-']; ['-'; '.'; '-']];;
 - : char list = ['-'; '-'; '-'; ' '; '-'; '.'; '-']
```

4.3 Impression

Écrivez la fonction qui va imprimer un mot en morse (au format char list list). Bien sur, pour rendre la chose lisible, il faut des espaces entre les lettres.

5 Level 3 : Les phrases !

5.1 Conversion (encore)

Écrivez la fonction qui à partir d'une phrase latine (liste de liste de caractères, oui oui.) renvoie la même phrase, sous forme de liste de liste de liste de caractères, en morse.

```
#sentence_to_morse [['t'; 'o']; ['m'; 'e']];;
 - : char list list list = [[['-']; ['-'; '-'; '-']]; [['-'; '-']; ['.']]]
```

5.2 char list list ...

Rassurez-vous, on ne va pa ajouter encore un "list" à la chose, mais en enlever une. Précédemment nous avons enlever un niveau de liste en séparant les lettres par des espaces. Ici, notre fonction va en plus séparer les mots par des '/'.

```
#sentence_to_single_list [[['-']; ['-'; '-'; '-']]; [['-'; '-']; ['.']]];;
- : char list = ['-'; ' '; '-'; '-'; '-'; '/'; '-'; '-'; ' '; '.'; '/']
```

5.3 Sans escales

Maintenant que nous savons comment réduire le niveau de listes en morse, on ne va pas s'arrêter en si bon chemin. Que diriez-vous de passer directement du latin au morse version "mono-liste" ? Faite la fonction qui part d'une liste de caractères latins pour arriver à un mot morse (avec les ' ' de séparation), puis une autre qui part d'une phrase (liste de liste de caractères) pour arriver à une liste en morse (avec les ' ' et les '/').

```
#to_single_morse ['S'; 'O'; 'S'] ;;
 - : char list = ['.'; '.'; '.'; ' '; '-'; '-'; '-'; ' '; '.'; '.'; '.']
#latin_sentence_to_single [['T'; 'O']; ['M'; 'E']] ;;
 - : char list = ['-'; ' '; '-'; '-'; '-'; '/'; '-'; '-'; ' '; '.'; '/']
```

6 Level 4 : Encoder moi tout ça !

Écrivez la, ou plutôt les fonctions qui vont vous permettre de passer d'une phrase latine à une phrase morse.

```
#latin_to_morse "Vive Les Listes" ;;
 - : string = "...- .. ...- ./.-.. . .../.-.. .. ... - . .../"
```

7 Level 5 : Décoder moi tout ça !

Écrivez la, ou plutôt les fonctions qui vont vous permettre de passer d'une phrase morse à une phrase latine.

```
#morse_to_latin "...- .. ...- ./.-.. . .../.-.. .. ... - . .../";;
 - : string = "VIVE LES LISTES"
```

8 Level 6 : Caml Spree

8.1 En rythme !

Écrivez une fonction qui affiche une phrase morse en rythme.
- Un '-' est aussi long que 3 '.'
- Les impulsions d'une même lettre sont séparées par un silence aussi long qu'un '.'
- Deux lettres sont séparées par un silence aussi long qu'un '-' (ou 3 '.')
- Deux mots sont séparés par un silence de 7 '.'

8.2 Bip bip bip !

Écrivez une fonction qui émet les sons correspondants à la séquence morse, le tout en rythme.

9 Level 7 : Et la genericite ?

Le morse c'est bien, mais avec tout ce qu'on vient de faire, ça serait bien de ne pas avoir à le refaire à chaque fois qu'on veut passer d'un alphabet à un autre... Heureusement, les fichiers sont là pour nous aider. Créez-vous un fichier de code, avec les correspondances Latin -> code, dans le format que vous souhaitez, puis grâce à quelques fonctions de lecture et correspondance dans un fichier, vous devriez pouvoir réutiliser une bonne partie de vos fonctions pour traduire l'alphabet latin dans n'importe quel autre, et inversement !