# Programmation fonctionnelle
# TD n° 5

Dans ce TD, on travaille sur des fichiers au format CSV (Comma-Separated Values). Un fichier CSV est un fichier texte dont le contenu représente un tableau. Chaque ligne du fichier correspond à une ligne du tableau. Les différentes valeurs composant une ligne sont séparées par des virgules.

Par exemple, le fichier `exemple.csv` dont le contenu est

```
Identifiant,Partiel,Projet,Final
,0.3,0.3,0.4
ab12345678,12,15,14
cd98765432,10,,9
ef24681357,11,13
```

correspond au tableau

| Identifiant | Partiel | Projet | Final |
|---|---|---|---|
| | 0.3 | 0.3 | 0.4 |
| ab12345678 | 12 | 15 | 14 |
| cd98765432 | 10 | | 9 |
| ef24681357 | 11 | 13 | |

Remarquez que les cases vides peuvent être représentées par l'absence d'une valeur (comme pour la première case de la deuxième ligne) ou tout simplement omises dans le fichier CSV si elle apparaît en fin de ligne (comme la dernière case).

## Exercice 1 (Manipulation des fichiers CSV)

On représente le contenu d'un fichier CSV par la liste de ces lignes. Une ligne est elle-même la liste de ses valeurs (des chaînes de caractères).

```
type csv_content = string list list
```

1. Définissez une fonction `input_lines : in_channel -> string list` qui renvoie la liste des lignes du canal d'entrée passé en paramètre.
2. Définissez une fonction `read_csv : string -> csv_content` qui lit et renvoie le contenu d'un fichier CSV dont le nom est passé en paramètre.
3. Définissez une fonction `join : string -> string list -> string` telle que `join sep l` concatène les éléments de `l` séparés par le séparateur `sep`.
4. Définissez une fonction `write_csv : csv_content -> string -> unit` telle que `csv_content c name` écrit `c` dans le fichier appelé `name` sous un format CSV.

5. Définissez une fonction `max_list : int list -> int list -> int list` qui renvoie une liste dont l'élément d'indice `i` est le maximum des éléments du même indice dans les listes passées en paramètres. Si une des listes est plus petite que l'autre, on considère que les éléments manquants sont égaux à 0.
6. Définissez une fonction `print_cell : int -> string -> unit` telle que `print_cell n str` affiche un espace, la chaîne `str` sur `n` caractères (on complète par des espaces si nécessaire), un espace et une barre. Par exemple, `print_cell 5 "foo"` affiche la chaîne `" foo   |"`.
7. Définissez une fonction `print_line : int list  -> string list -> unit` qui affiche le contenu d'une ligne. La première liste correspond aux tailles des colonnes et la deuxième au contenu des cellules. Par exemple, `print_line [5; 3] ["foo"; "bar"]` affiche la chaîne `"| foo   | bar |"`.
8. Définissez une fonction `print_csv : csv_content -> unit` qui affiche le contenu d'un fichier CSV comme proposé dans l'exemple introductif.

## Exercice 2 (Analyse d'un fichier CSV)

On cherche à analyser des fichiers CSV dans un format particulier. La première ligne contient des intitulés. La deuxième ligne contient des coefficients (sauf pour la première cellule) correspondant à la pondération d'un examen pour le calcul de la moyenne d'une UE. Les lignes suivantes correspondent à des étudiants composés d'un identifiant et d'un certain nombre de notes. On cherche à obtenir le contenu d'un tel fichier comme un enregistrement `data`.

```
type student = {
  id : string;
  grades : float option list; (* None correspond à ABI *)
}

type data = {
  header : string list;
  coeff : float list;
  students : student list;
}
```

1. Définissez une exception `Invalid_format` qui sera levée dès que le fichier analysé ne correspond pas aux spécifications.
2. Définissez une fonction `parse_coeff : string list -> float list` qui prend en paramètre la ligne correspondant aux coefficients et les convertit en flottant.
3. Définissez une fonction `grades_from_strings : string list -> float option list` qui prend en paramètre la liste des notes et les convertit en flottants si possible. Toute note que l'on ne peut convertir en flottant devient la valeur `None`.
4. Définissez une fonction `normalize_grade : int -> float option list -> float option list` telle que `normalize_grade n l` renvoie la liste `l` complétée par des valeurs `None` pour qu'elle soit de taille `n`. Si la liste a une taille supérieure à `n`, on lève une exception `Invalid_format`.
5. Définissez une fonction `parse_student : int -> string list -> student` qui prend en paramètres un nombre de notes attendues et une ligne correspondant à un étudiant. Elle renvoie la représentation de l'étudiant associée.
6. Définissez une fonction `parse_csv : csv_content -> data` qui analyse le contenu d'un fichier CSV et en produit la représentation. Le nombre de notes attendues pour les étudiants est déduit du nombre de coefficients sur la deuxième ligne du fichier.

## Exercice 3 (Modification d'un fichier CSV)

On cherche maintenant à calculer la moyenne des étudiants présents dans un fichier CSV. Il faudra définir une fonction `process_file : string -> string -> unit` qui charge le contenu d'un fichier CSV, ajoute une colonne qui contient la moyenne de chaque étudiant, et l'écrit dans un nouveau fichier.

1. Définissez une fonction `moyenne : float list -> float option list -> float` qui prend en paramètres une liste des coefficients et une liste de note pour renvoyer la moyenne pondérée de ces notes.
2. Définissez une fonction `process_student : float list -> student -> string list` qui prend en paramètres une liste de coefficients et un étudiant. Elle renvoie la ligne correspondant à l'étudiant. On y trouve son identifiant, ses notes (les valeurs `None` deviennent `"ABI"`) et sa moyenne.
3. Définissez la fonction `process_file` telle que `process_file name_in name_out` lit le fichier `name_in` et écrit dans le fichier `name_out`.

Par exemple, le code suivant

```
process_file "exemple.csv" "moyenne.csv";;
print_csv (read_csv "moyenne.csv");;
```

affiche

```
-------------------------------------------------
| Identifiant | Partiel | Projet | Final | Moyenne |
-------------------------------------------------
|             | 0.3     | 0.3    | 0.4   |         |
-------------------------------------------------
| ab12345678  | 12.     | 15.    | 14.   | 13.7    |
-------------------------------------------------
| cd98765432  | 10.     | ABI    | 9.    | 6.6     |
-------------------------------------------------
| ef24681357  | 11.     | 13.    | ABI   | 7.2     |
-------------------------------------------------
```