

TP n° 5

Consignes les exercices ou questions marqués d'un \star devront être rédigés sur papier (afin de se préparer aux épreuves écrites de l'examen). En particulier, il est recommandé d'être dans les mêmes conditions qu'en examen : pas de document ni de calculatrice. Tous les TPs se font sous Linux.

1 Une base de données de films

Le but du TP est d'utiliser des itérateurs sur les listes ainsi que des fonctions récursives sur ces dernières pour effectuer de petites requêtes sur une base de données de films, stockée dans un fichier.

1. Parcourir rapidement le fichier `movies.csv` disponible sur la page du cours. Ce dernier contient une liste de films, chacun sur une ligne. Chaque film est composé de champs séparés par le caractère « ; ». Les champs contiennent respectivement un identifiant pour le film (entier), le titre du film (chaîne de caractères ne contenant pas de « ; »), l'année de sortie du film (un entier), la durée du film en minutes (un entier) et le rang de ce film dans un classement des meilleurs films (un entier). Tous les entiers sont supérieurs ou égaux à 0.
2. \star Que font les lignes 1 à 7 du fichier ?

```
1 type movie = {
2   id : int;
3   title : string;
4   year : int;
5   runtime : int;
6   rank : int
7 }
```

3. \star Que fait la ligne 9 du fichier ?

```
1 type res = Movie of movie | Invalid | Eof
```

4. \star en OCaml, le type `in_channel` est le type des descripteurs de fichiers ouverts en lecture. Sachant que la fonction `input_line` permet de lire la ligne suivante dans un fichier ouvert en lecture, qu'elle est de type `in_channel -> string`, qu'elle peut lever l'exception `End_of_file` si on a fini de lire le fichier et que la fonction `int_of_string` peut lever une exception si la chaîne donnée n'est pas un entier :
 - donner le type de la fonction `input_movie`
 - dire ce que fait cette fonction

```
1 let input_movie in_c =
2   try
3     let s = input_line in_c in
4     match String.split_on_char ';' s with
5     [ s_id; title; s_year; s_runtime ; s_rank ] ->
6       Movie ({
7         id = int_of_string s_id;
8         title = title;
9         year = int_of_string s_year;
10        runtime = int_of_string s_runtime;
11        rank = int_of_string s_rank;
12      })
```

```

13 | _ -> Invalid
14
15 with
16 End_of_file -> Eof
17 | _ -> Invalid

```

5. * Sachant que la fonction `open_in` est de type `string -> in_channel`, donner le type de la fonction `load_movies` et dire ce qu'elle fait. Quel type pouvez vous déduire pour la fonction `close_in`.

```

1 let load_movies f =
2   let in_c = open_in f in
3   let rec loop in_c acc =
4     match input_movie in_c with
5     | Eof -> acc
6     | Invalid -> loop in_c acc
7     | Movie m -> loop in_c (m :: acc)
8   in
9   let res = loop in_c [] in
10  close_in in_c;
11  res
12 ;;

```

6. * Que fait la ligne 43?

```

1 let movies = load_movies "movies.csv"

```

7. Écrire une fonction `pr_movie : movie -> unit` qui affiche un film dans la console au format suivant :
- ```
{ id=2004; title="The Good,the Bad and the Ugly"; year=1966; runtime=190; rank=5 }
```
8. Écrire une fonction `pr_movies : movie list -> unit` qui affiche les films de la liste donnée ligne par ligne. La fonction doit utiliser un itérateur du module `List`.
9. Écrire une fonction `moviesTop10 : movie list -> movie list` qui renvoie la liste des films dont le rang est inférieur ou égal à 10. La fonction doit utiliser un itérateur du module `List`. Afficher ces résultats dans la console.
10. Écrire une fonction `movies1980 : movie list -> movie list` qui renvoie la liste des films des années 80 (année comprise entre 1980 et 1989). La fonction doit utiliser un itérateur du module `List`. Afficher ces résultats dans la console.
11. Écrire une fonction `movie_titles : movie list -> string list` qui renvoie la liste des titres des films. La fonction doit utiliser un itérateur du module `List`. Afficher ces résultats dans la console.
12. Écrire une fonction `max_id : movie list -> int` qui renvoie le plus grand identifiant de film de la liste ou 0 si la liste est vide. La fonction doit utiliser l'itérateur `List.fold_left`. Afficher ce résultat dans la console.
13. Écrire une fonction `average_runtime : movie list -> float` qui renvoie la moyenne des durées des films (on suppose que la liste donnée en argument n'est pas vide). La fonction doit utiliser l'itérateur `List.fold_left` et ne faire qu'un seul parcours de liste (en particulier ne pas utiliser la fonction `List.length`). Afficher ce résultat dans la console.
14. Écrire une fonction `average_by_year : movie list -> (int * float) list` qui calcule la moyenne des durées des films pour chaque année présente dans la base.

L'algorithme ici est le suivant :

- trier la liste de films par année (croissantes ou décroissantes, peu importe)
- Écrire une fonction récursive `loop` qui parcourt la liste triée et maintient un accumulateur. Ce dernier est la liste des paires (*anne, liste des films de cette anne*).
  - si la liste des films est vide, renvoyer l'accumulateur
  - si l'accumulateur est vide, créer un nouveau couple avec l'année et la liste contenant le film et le placer dans l'accumulateur vide, puis se rappeler récursivement

- sinon, comparer l'année du film courant et l'année en tête d'accumulateur. S'ils sont égaux, ajouter le film a la liste des films en tête de liste. Sinon rajouter la nouvelle année comme une nouvelle paire dans l'accumulateur. Dans tous les cas se rappeler récursivement avec le nouvel accumulateur
- Une fois la liste de paires ainsi obtenue, appeler la fonction `average_runtime` sur chaque seconde composante de chaque paire de la liste.

15. Constater que la fonction `average_by_year` peut être généralisée en une fonction

`group_by : ('a -> 'b) -> 'a list -> ('b * 'a list) list`

Cette dernière prend en argument : une fonction permettant d'extraire d'un élément une valeur clé (par exemple l'année), une liste de valeurs (par exemple des films) et qui renvoie une liste de couples clé, liste des éléments ayant cette valeur de clé. Écrire une telle fonction et utiliser en plus un `List.fold_left` à la place de la fonction récursive `loop`.

16. Utiliser la fonction `group_by` pour définir une fonction `average_by_decade` qui calcule les moyennes des durées de films pour chaque décennie.