Programmation fonctionnelle

Fiche de TP 6

L3 Informatique 2020-2021

Système de fichiers

⚠⚠⚠
Ce TP ressemble dans son déroulement, sa présentation
et le type des questions posées à l'**examen final** à venir.
⚠⚠⚠

1 Échauffement et conseils généraux

Exercice 1. (**Arbres généraux**) — *Environ 10 minutes*

Dans cet exercice, on utilisera le type suivant pour les arbres généraux (0, 1, 2 ou plus de fils pour chaque nœud).

```
type tree =
   |Leaf of int
   |Node of char * tree list
```

Ces arbres ont deux types de nœuds :

— soit des nœuds internes contenant des caractères (ils peuvent avoir 0 fils) ;

— soit des feuilles contenant des entiers.

On utilisera l'arbre suivant comme exemple.

```
let example_tree =
   Node ('a',
   [Node ('b', [Leaf 1; Leaf 2; Leaf 3]);
       Node ('c', [Node ('d', [Leaf 4; Leaf 5]); Node ('e', [Leaf 6])]);
       Leaf 7;
       Node ('f', [Node ('g', [Leaf 8]); Leaf 9])])
```

1. Dessiner (sur une feuille !) l'arbre donné en exemple.
2. Écrire la fonction `tree_count_nodes` qui renvoie le nombre de nœuds internes dans un arbre passé en paramètre.
   (*Indication : il peut être utile d'écrire une fonction auxiliaire qui prend en paramètre une liste d'arbres*)
3. Écrire la fonction `tree_list_leaves` qui renvoie la liste des feuilles d'un arbre passé en paramètre.

```
# tree_count_nodes example_tree;;
- : int = 7

# tree_list_leaves example_tree;;
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9]
```

⋆⋆⋆

Le but de ce TP est de développer un ensemble de fonctions pour manipuler un système de fichiers simplifié. Nous manipulerons des *répertoires* (« folder » en anglais) contenant des *fichiers* (« file » en anglais) et des répertoires. Pour simplifier nous ne considérerons dans la suite que six types de fichier différents : `pdf`, `doc`, `jpg`, `png`, `avi` et `mkv`.

Voici quelques remarques et conseils avant de passer à la suite.

— Les types des fonctions demandées sont toujours donnés dans les questions.

— Vous trouverez un exemple complet d'utilisation de ces fonctions à la fin de ce sujet.

— Pensez à utiliser les fonctions des modules `List` et `String`. En particulier, les fonctions

```
List.map : ('a -> 'b) -> 'a list -> 'b list = <fun>
List.filter : ('a -> bool) -> 'a list -> 'a list = <fun>
String.compare : fs_item_name -> fs_item_name -> int = <fun>
```

seront particulièrement utiles.

— Dans certaines fonctions, le type inféré par CAML utilisera le type le plus général `String.t` à la place du type spécifique `fs_item_name` (qui est justement un alias sur les chaînes de caractères). Vous ne serez pas pénalisés sur **ce** point.

— Pour vos tests, utilisez le fichier `test_file_system.ml` qui contient les tests présentés à la fin de ce sujet.

2 Système de Fichiers

Nous utiliserons tout au long de ce TP les types CAML suivants :

```
(* file system item name alias *)
type fs_item_name = string

(* file system folder *)
(* a folder is simply described by its name *)
type fs_folder = Folder of fs_item_name

(* file system file type *)
type fs_file_type =
    |PDF (* portable document format *)
    |DOC (* microsoft document *)
    |PNG (* portable network graphics *)
    |JPG (* joint photographic experts group, *)
    |AVI (* audio video interleave *)
    |MKV (* matroska video *)

(* file system file size alias *)
type fs_file_size = int

(* file system file *)
(* a file is described by its name, its type and its size *)
type fs_file = File of fs_item_name * fs_file_type * fs_file_size
```

On utilise le type récursif

```
(* file system item *)
(* a file system item is either a folder (containing items) or a file *)
type fs_item =
    |FolderItem of fs_folder * fs_item list
    |FileItem of fs_file
```

pour définir un système de fichiers. Ainsi, un *élément dans un système de fichiers* (`fs_item`) est

- soit un répertoire (`FolderItem`) qui contient notamment une liste de systèmes de fichiers ;
- soit un fichier (`FileItem`).

3 Fonctions sur les listes de fichiers

Dans cette partie, vous commencerez par écrire deux fonctions qui permettent de lister respectivement les noms des fichiers et les noms des répertoires d'un système de fichiers. Puis vous devrez écrire des fonctions pour manipuler les listes ainsi formées.

Question 1. Écrire la fonction

```
val files : fs_item -> fs_file list = <fun>
```

qui prend en paramètre un système de fichiers (`fs_item`) et renvoie la liste de tous les fichiers (`fs_file list`) qu'il contient (où que se trouvent ces fichiers dans le système de fichiers). L'ordre des fichiers dans la liste renvoyée n'a pas d'importance.

Question 2. Écrire la fonction

```
val folders : fs_item -> fs_folder list = <fun>
```

qui prend en paramètre un système de fichiers (`fs_item`) et renvoie la liste de tous les répertoires (`fs_folder list`) qu'il contient (où que se trouvent ces répertoires dans le système de fichiers). L'ordre des répertoires dans la liste renvoyée n'a pas d'importance.

Question 3. Écrire les trois prédicats

```
val is_image : fs_file -> bool = <fun>
val is_movie : fs_file -> bool = <fun>
val is_document : fs_file -> bool = <fun>
```

qui testent respectivement si un objet de type `fs_file` est une image (type de fichier `JPG` ou `PNG`), un film (type de fichier `AVI` ou `MKV`) ou un document (type de fichier `DOC` ou `PDF`).

Question 4. Écrire les trois fonctions

```
val images : fs_item -> fs_file list = <fun>
val movies : fs_item -> fs_file list = <fun>
val documents : fs_item -> fs_file list = <fun>
```

qui renvoient respectivement une liste de toutes les images, tous les films ou tous les documents (`fs_file list` dans les trois cas) présents dans le système de fichiers passé en paramètre (vous éviterez ici au maximum la duplication de code). L'ordre des éléments dans les listes renvoyées n'a pas d'importance.

Question 5. Écrire la fonction

```
val rec_search_list : fs_file list -> fs_item_name -> fs_file list = <fun>
```

qui prend en paramètre une liste de fichiers (`fs_file list`) et une chaîne de caractères `name` et renvoie tous les fichiers de la liste ayant pour nom `name`, indépendamment du type (`pdf`, `doc`, `jpg`, `png`, `avi` ou `mkv`) du fichier. Cette fonction devra être **récursive non terminale** (sans accumulateur).

Question 6. Écrire la fonction

```
val tail_rec_search_list : fs_file list -> fs_item_name -> fs_file list = <fun>
```

qui fait la même chose que `rec_search_list`, mais qui est cette fois-ci **récursive terminale** (donc avec une fonction locale auxiliaire qui utilise un accumulateur).

Question 7. Écrire la fonction

```
val not_rec_search_list : fs_file list -> fs_item_name -> fs_file list = <fun>
```

qui fait la même chose que `rec_search_list`, mais **sans utiliser la construction** `let rec` (donc avec une fonction d'ordre supérieur).

Question 8. Utiliser la curryfication pour écrire la fonction

```
val search_documents : fs_item -> fs_item_name -> fs_item list = <fun>
```

qui prend en paramètre un système de fichiers et renvoie une fonction permettant de chercher tous les documents ayant un nom donné (en paramètre), dans le système de fichiers (où que se trouvent ces documents dans le système de fichiers).

Question 9. Nous avons développé plusieurs fonctions de recherche dans les questions précédentes et nous souhaitons pouvoir les utiliser indifféremment dans la fonction `search_documents`. Utiliser alors la curryfication pour écrire la fonction

```
val search_documents_fun : (fs_file list -> 'a) -> fs_item -> 'a = <fun>
```

qui prend en paramètre une fonction de recherche (comme la fonction `rec_search_list` ou `tail_rec_search_list` ou encore `not_rec_search_list`) et un système de fichiers (`fs_item`) et renvoie une fonction permettant de chercher tous les documents ayant un nom donné (en paramètre), dans le système de fichiers.

Question 10. Utiliser une **fonction d'ordre supérieur** (et donc pas de construction `let rec`) pour écrire la fonction

```
val size_images : fs_item -> int = <fun>
```

qui prend en paramètre un système de fichiers (`fs_item`) et renvoie la somme des tailles de toutes les images qu'il contient.

4 Fonctions sur les systèmes de fichiers arborescents

⚠⚠

Dans cette partie, vous devez éviter de parcourir plusieurs fois les items d'un système de fichiers, lorsque c'est possible. Cela signifie notamment que vous ne pouvez pas commencer par lister tout ou une partie des items d'un système de fichiers avant de leur appliquer le traitement demandé.

⚠⚠

Question 11. Écrire la fonction d'ordre supérieur

```
val fs_filter : (fs_item -> bool) -> fs_item -> fs_item_name list = <fun>
```

qui prend en paramètre un prédicat sur les items du système de fichiers (`fs_item -> bool`) et un système de fichiers (`fs_item`) et renvoie une liste des noms de tous les fichiers et répertoires (`fs_item_name list`) qui vérifient le prédicat (où que se trouvent ces fichiers et répertoires dans le système de fichiers).

Question 12. En utilisant la fonction `fs_filter`, écrire

1. la fonction

   ```
   val item_names_with_large6_name : fs_item -> fs_item_name list = <fun>
   ```

   qui renvoie la liste de tous les noms de fichier ou répertoire (`fs_item_name list`) d'un système de fichiers dont le nom contient au moins six caractères (où que se trouvent ces fichiers et répertoires dans le système de fichiers) ;

2. la fonction

   ```
   val digit_in_names : fs_item -> fs_item_name list = <fun>
   ```

   qui renvoie la liste de tous les noms de fichier ou répertoire (`fs_item_name list`) dont le nom contient au moins un chiffre dans un système de fichiers (où que se trouvent ces fichiers et répertoires dans le système de fichiers).

Question 13. Écrire la fonction

```
val full_paths : fs_item -> fs_item_name list = <fun>
```

qui prend en paramètre un système de fichiers `fs` (`fs_item`) et renvoie la liste de tous les chemins complets (depuis la racine de `fs`) vers tous ses fichiers et répertoires.

Indice : pensez à utiliser la fonction `concat : string -> string list -> string`.

Question 14. Écrire la fonction

```
val name_with_ext : fs_item -> fs_item_name = <fun>
```

qui prend en paramètre un système de fichiers `fs` (`fs_item`) et qui renvoie soit le nom du répertoire `fs` soit le nom du fichier `fs` suffixé par son type.

Question 15. Écrire la fonction

```
val no_two_identical_names : fs_item_name list -> bool = <fun>
```

qui prend en paramètre une liste de chaînes de caractères (`fs_item_name list`) et qui teste si la liste ne contient pas deux fois la même chaîne de caractères. Vous **ne** pouvez **pas** supposer que la liste initiale est préalablement triée.

Question 16. En utilisant les fonctions `name_with_ext` et `no_two_identical_names`, écrire la fonction

```
val check : fs_item -> bool = <fun>
```

qui prend en paramètre un système de fichiers `fs` (`fs_item`) et vérifie s'il est bien formé, c'est-à-dire que la racine est bien un répertoire et qu'aucun dossier de `fs` ne contient deux fichiers de même nom et type ou deux répertoires de même nom. Notez que le type distingue deux fichiers de même nom mais de type différent, de sorte que les fichiers `f.doc` et `f.pdf` sont par exemple distincts.

Question 17. Écrire la fonction

```
val du : fs_item -> (fs_item_name * fs_file_size) list = <fun>
```

qui prend en paramètre un système de fichiers `fs` (`fs_item`) et renvoie une liste de couples représentant le nom (avec l'extension quand il s'agit d'un fichier) et la taille de tous les fichiers et répertoires de `fs`. Pour un répertoire, la taille est la somme des tailles de tous les items qu'il contient. Pour cette question, en particulier, vous ne devez parcourir le système de fichiers qu'une seule fois.

5 Exemple complet

```
(* toy file system my_fs *)
val my_fs : fs_item =
 FolderItem (Folder "root",
  [FolderItem (Folder "Documents",
    [FileItem (File ("doc 1", DOC, 32)); FileItem (File ("doc 2", DOC, 64));
     FileItem (File ("doc 3", PDF, 1024));
     FileItem (File ("doc 4", PDF, 2048));
     FolderItem (Folder "2015",
      [FileItem (File ("sujet tp note", PDF, 512));
       FileItem (File ("notes tp", DOC, 64))]);
     FolderItem (Folder "2016",
      [FileItem (File ("sujet tp note", PDF, 512));
       FileItem (File ("notes tp", DOC, 64))])]);
   FileItem (File ("Documents", PDF, 512));
   FileItem (File ("config", DOC, 28));
   FolderItem (Folder "Downloads",
    [FileItem (File ("doc 1", DOC, 32)); FileItem (File ("doc 2", DOC, 64));
     FileItem (File ("doc 1", PDF, 1024));
     FileItem (File ("doc 2", PDF, 2048))]);
   FolderItem (Folder "Movies",
    [FolderItem (Folder "Rocky 1",
      [FileItem (File ("Rocky 1", MKV, 4294967296));
       FileItem (File ("Rocky 1 - subtitle fr", DOC, 4096));
       FileItem (File ("Rocky 1 - subtitle en", DOC, 4096))]);
```

```
    FolderItem (Folder "Jaws 2",
     [FileItem (File ("Jaws 2", AVI, 16777216));
      FileItem (File ("Jaws 2 - subtitle fr", DOC, 4096))]);
    FolderItem (Folder "Alien 3",
     [FileItem (File ("Alien 3", MKV, 4294967296))]);
    FileItem (File ("Seven", AVI, 1024));
    FileItem (File ("seen movies", DOC, 64))]);
  FolderItem (Folder "Pictures",
   [FileItem (File ("Martine fait du chameau 1", PNG, 2048));
    FileItem (File ("Martine fait du chameau 2", JPG, 4096));
    FolderItem (Folder "Photos 2015",
     [FileItem (File ("description 2015", DOC, 256));
      FileItem (File ("Martine au zoo 1", JPG, 2048));
      FileItem (File ("Martine au zoo 2", JPG, 2048));
      FileItem (File ("Martine au zoo 3", PNG, 2048))]);
    FolderItem (Folder "Photos 2016",
     [FileItem (File ("description 2016", DOC, 512));
      FileItem (File ("Martine mange une pomme 1", JPG, 2048));
      FileItem (File ("Martine mange une pomme 2", JPG, 2048));
      FileItem (File ("Martine mange une pomme 3", PNG, 2048));
      FileItem (File ("Martine mange une pomme 4", PNG, 2048))])])])

(* toy file system my_fs2 *)
val my_fs2 : fs_item =
 FolderItem (Folder "root",
  [FolderItem (Folder "Documents",
    [FileItem (File ("doc 1", DOC, 32));
     FileItem (File ("doc 2", DOC, 64));
     FileItem (File ("doc 3", PDF, 1024));
     FileItem (File ("doc 4", PDF, 2048))]);
   FileItem (File ("Documents", DOC, 28))])

(* toy file system my_fs3 *)
val my_fs3 : fs_item =
 FolderItem (Folder "root",
  [FolderItem (Folder "Documents",
    [FolderItem (Folder "Documents",
      [FileItem (File ("doc 1", DOC, 32));
       FileItem (File ("doc 2", DOC, 64))]);
     FileItem (File ("doc 1", DOC, 32));
     FileItem (File ("doc 2", DOC, 64))]);
   FileItem (File ("Downloads", DOC, 28))])

(* toy file system my_fs4 *)
val my_fs4 : fs_item =
 FolderItem (Folder "root",
  [FolderItem (Folder "Documents",
    [FileItem (File ("doc 1", DOC, 32));
     FileItem (File ("doc 2", DOC, 64));
     FileItem (File ("doc 1", PDF, 1024));
     FileItem (File ("doc 2", PDF, 2048))]);
```

```
    FolderItem (Folder "Documents", [])])

(* toy file system my_fs5 *)
val my_fs5 : fs_item =
 FolderItem (Folder "root",
  [FolderItem (Folder "Documents",
    [FileItem (File ("doc 1", DOC, 32));
     FileItem (File ("doc 1", DOC, 64));
     FileItem (File ("doc 3", PDF, 1024));
     FileItem (File ("doc 4", PDF, 2048))]);
   FileItem (File ("Documents", DOC, 28))])

(* toy file system my_fs6 *)
val my_fs6 : fs_item =
 FolderItem (Folder "root",
  [FolderItem (Folder "Documents",
    [FileItem (File ("doc 1", DOC, 32));
     FileItem (File ("doc 1", DOC, 64))]);
   FileItem (File ("AutresDocuments", DOC, 28))])

(* File examples *)
val test1 : fs_file = File ("file 1", DOC, 32)
val test2 : fs_file = File ("file 2", PNG, 512)
val test3 : fs_file = File ("file 3", AVI, 4294967296)

(* testing files function *)
files my_fs;;
- : fs_file list =
[File ("Martine mange une pomme 4", PNG, 2048);
 File ("Martine mange une pomme 3", PNG, 2048);
 File ("Martine mange une pomme 2", JPG, 2048);
 File ("Martine mange une pomme 1", JPG, 2048);
 File ("description 2016", DOC, 512); File ("Martine au zoo 3", PNG, 2048);
 File ("Martine au zoo 2", JPG, 2048); File ("Martine au zoo 1", JPG, 2048);
 File ("description 2015", DOC, 256);
 File ("Martine fait du chameau 2", JPG, 4096);
 File ("Martine fait du chameau 1", PNG, 2048);
 File ("seen movies", DOC, 64); File ("Seven", AVI, 1024);
 File ("Alien 3", MKV, 4294967296); File ("Jaws 2 - subtitle fr", DOC, 4096);
 File ("Jaws 2", AVI, 16777216); File ("Rocky 1 - subtitle en", DOC, 4096);
 File ("Rocky 1 - subtitle fr", DOC, 4096);
 File ("Rocky 1", MKV, 4294967296); File ("doc 2", PDF, 2048);
 File ("doc 1", PDF, 1024); File ("doc 2", DOC, 64); File ("doc 1", DOC, 32);
 File ("config", DOC, 28); File ("Documents", PDF, 512);
 File ("notes tp", DOC, 64); File ("sujet tp note", PDF, 512);
 File ("notes tp", DOC, 64); File ("sujet tp note", PDF, 512);
 File ("doc 4", PDF, 2048); File ("doc 3", PDF, 1024);
 File ("doc 2", DOC, 64); File ("doc 1", DOC, 32)]

(* testing folders function *)
folders my_fs;;
```

```
- : fs_folder list =
[Folder "Photos 2016"; Folder "Photos 2015"; Folder "Pictures";
 Folder "Alien 3"; Folder "Jaws 2"; Folder "Rocky 1"; Folder "Movies";
 Folder "Downloads"; Folder "2016"; Folder "2015"; Folder "Documents";
 Folder "root"]

(* testing is_image, is_movies and is_document functions *)
is_image test1, is_image test2, is_image test3;;
- : bool * bool * bool = (false, true, false)
is_movie test1, is_movie test2, is_movie test3;;
- : bool * bool * bool = (false, false, true)
is_document test1, is_document test2, is_document test3;;
- : bool * bool * bool = (true, false, false)

(* testing images function *)
images my_fs;;
- : fs_file list =
[File ("Martine mange une pomme 4", PNG, 2048);
 File ("Martine mange une pomme 3", PNG, 2048);
 File ("Martine mange une pomme 2", JPG, 2048);
 File ("Martine mange une pomme 1", JPG, 2048);
 File ("Martine au zoo 3", PNG, 2048); File ("Martine au zoo 2", JPG, 2048);
 File ("Martine au zoo 1", JPG, 2048);
 File ("Martine fait du chameau 2", JPG, 4096);
 File ("Martine fait du chameau 1", PNG, 2048)]

(* testing movies function *)
movies my_fs;;
- : fs_file list =
[File ("Seven", AVI, 1024); File ("Alien 3", MKV, 4294967296);
 File ("Jaws 2", AVI, 16777216); File ("Rocky 1", MKV, 4294967296)]

(* testing document function *)
documents my_fs;;
- : fs_file list =
[File ("description 2016", DOC, 512); File ("description 2015", DOC, 256);
 File ("seen movies", DOC, 64); File ("Jaws 2 - subtitle fr", DOC, 4096);
 File ("Rocky 1 - subtitle en", DOC, 4096);
 File ("Rocky 1 - subtitle fr", DOC, 4096); File ("doc 2", PDF, 2048);
 File ("doc 1", PDF, 1024); File ("doc 2", DOC, 64); File ("doc 1", DOC, 32);
 File ("config", DOC, 28); File ("Documents", PDF, 512);
 File ("notes tp", DOC, 64); File ("sujet tp note", PDF, 512);
 File ("notes tp", DOC, 64); File ("sujet tp note", PDF, 512);
 File ("doc 4", PDF, 2048); File ("doc 3", PDF, 1024);
 File ("doc 2", DOC, 64); File ("doc 1", DOC, 32)]

(* testing rec_search_list function *)
rec_search_list (files my_fs) "notes td";;
- : fs_file list = []
rec_search_list (files my_fs) "notes tp";;
- : fs_file list = [File ("notes tp", DOC, 64); File ("notes tp", DOC, 64)]
```

```
(* testing tail_rec_search_list function *)
tail_rec_search_list (files my_fs) "notes td";;
- : fs_file list = []
tail_rec_search_list (files my_fs) "notes tp";;
- : fs_file list = [File ("notes tp", DOC, 64); File ("notes tp", DOC, 64)]

(* testing not_rec_search_list function *)
not_rec_search_list (files my_fs) "notes td";;
- : fs_file list = []
not_rec_search_list (files my_fs) "notes tp";;
- : fs_file list = [File ("notes tp", DOC, 64); File ("notes tp", DOC, 64)]

(* testing search_documents function *)
search_documents my_fs "notes td";;
- : fs_file list = []
search_documents my_fs "notes tp";;
- : fs_file list = [File ("notes tp", DOC, 64); File ("notes tp", DOC, 64)]
search_documents my_fs "Rocky 1";;
- : fs_file list = []

(* testing size_images function *)
size_images my_fs;;
- : int = 20480

(* testing item_names_with_large6_name *)
item_names_with_large6_name my_fs;;
- : fs_item_name list =
["Documents"; "sujet tp note"; "notes tp"; "sujet tp note"; "notes tp";
 "Documents"; "config"; "Downloads"; "Movies"; "Rocky 1"; "Rocky 1";
 "Rocky 1 - subtitle fr"; "Rocky 1 - subtitle en"; "Jaws 2"; "Jaws 2";
 "Jaws 2 - subtitle fr"; "Alien 3"; "Alien 3"; "seen movies"; "Pictures";
 "Martine fait du chameau 1"; "Martine fait du chameau 2"; "Photos 2015";
 "description 2015"; "Martine au zoo 1"; "Martine au zoo 2";
 "Martine au zoo 3"; "Photos 2016"; "description 2016";
 "Martine mange une pomme 1"; "Martine mange une pomme 2";
 "Martine mange une pomme 3"; "Martine mange une pomme 4"]

(* testing item_names_with_digit_in_name function *)
item_names_with_digit_in_name my_fs;;
- : fs_item_name list =
["doc 1"; "doc 2"; "doc 3"; "doc 4"; "2015"; "2016"; "doc 1"; "doc 2";
 "doc 1"; "doc 2"; "Rocky 1"; "Rocky 1"; "Rocky 1 - subtitle fr";
 "Rocky 1 - subtitle en"; "Jaws 2"; "Jaws 2"; "Jaws 2 - subtitle fr";
 "Alien 3"; "Alien 3"; "Martine fait du chameau 1";
 "Martine fait du chameau 2"; "Photos 2015"; "description 2015";
 "Martine au zoo 1"; "Martine au zoo 2"; "Martine au zoo 3"; "Photos 2016";
 "description 2016"; "Martine mange une pomme 1";
 "Martine mange une pomme 2"; "Martine mange une pomme 3";
 "Martine mange une pomme 4"]

```

```
(* testing full_path function *)
full_paths my_fs;;
- : string list =
["/root"; "/root/Documents"; "/root/Documents/doc␣1";
 "/root/Documents/doc␣2"; "/root/Documents/doc␣3"; "/root/Documents/doc␣4";
 "/root/Documents/2015"; "/root/Documents/2015/sujet␣tp␣note";
 "/root/Documents/2015/notes␣tp"; "/root/Documents/2016";
 "/root/Documents/2016/sujet␣tp␣note"; "/root/Documents/2016/notes␣tp";
 "/root/Documents"; "/root/config"; "/root/Downloads";
 "/root/Downloads/doc␣1"; "/root/Downloads/doc␣2"; "/root/Downloads/doc␣1";
 "/root/Downloads/doc␣2"; "/root/Movies"; "/root/Movies/Rocky␣1";
 "/root/Movies/Rocky␣1/Rocky␣1";
 "/root/Movies/Rocky␣1/Rocky␣1␣-␣subtitle␣fr";
 "/root/Movies/Rocky␣1/Rocky␣1␣-␣subtitle␣en"; "/root/Movies/Jaws␣2";
 "/root/Movies/Jaws␣2/Jaws␣2"; "/root/Movies/Jaws␣2/Jaws␣2␣-␣subtitle␣fr";
 "/root/Movies/Alien␣3"; "/root/Movies/Alien␣3/Alien␣3";
 "/root/Movies/Seven"; "/root/Movies/seen␣movies"; "/root/Pictures";
 "/root/Pictures/Martine␣fait␣du␣chameau␣1";
 "/root/Pictures/Martine␣fait␣du␣chameau␣2"; "/root/Pictures/Photos␣2015";
 "/root/Pictures/Photos␣2015/description␣2015";
 "/root/Pictures/Photos␣2015/Martine␣au␣zoo␣1";
 "/root/Pictures/Photos␣2015/Martine␣au␣zoo␣2";
 "/root/Pictures/Photos␣2015/Martine␣au␣zoo␣3"; "/root/Pictures/Photos␣2016";
 "/root/Pictures/Photos␣2016/description␣2016";
 "/root/Pictures/Photos␣2016/Martine␣mange␣une␣pomme␣1";
 "/root/Pictures/Photos␣2016/Martine␣mange␣une␣pomme␣2";
 "/root/Pictures/Photos␣2016/Martine␣mange␣une␣pomme␣3";
 "/root/Pictures/Photos␣2016/Martine␣mange␣une␣pomme␣4"]

(* testing check function *)
check my_fs, check my_fs2, check my_fs3, check my_fs4, check my_fs5, check my_fs6;;
- : bool * bool * bool * bool * bool * bool = (true, true, true, false, false, false)

(* testing du function *)
du my_fs;;
- : (fs_item_name * fs_file_size) list =
[("root", 8606754460); ("Documents", 4320); ("doc␣1.doc", 32);
 ("doc␣2.doc", 64); ("doc␣3.pdf", 1024); ("doc␣4.pdf", 2048); ("2015", 576);
 ("sujet␣tp␣note.pdf", 512); ("notes␣tp.doc", 64); ("2016", 576);
 ("sujet␣tp␣note.pdf", 512); ("notes␣tp.doc", 64); ("Documents.pdf", 512);
 ("config.doc", 28); ("Downloads", 3168); ("doc␣1.doc", 32);
 ("doc␣2.doc", 64); ("doc␣1.pdf", 1024); ("doc␣2.pdf", 2048);
 ("Movies", 8606725184); ("Rocky␣1", 4294975488);
 ("Rocky␣1.mkv", 4294967296); ("Rocky␣1␣-␣subtitle␣fr.doc", 4096);
 ("Rocky␣1␣-␣subtitle␣en.doc", 4096); ("Jaws␣2", 16781312);
 ("Jaws␣2.avi", 16777216); ("Jaws␣2␣-␣subtitle␣fr.doc", 4096);
 ("Alien␣3", 4294967296); ("Alien␣3.mkv", 4294967296); ("Seven.avi", 1024);
 ("seen␣movies.doc", 64); ("Pictures", 21248);
 ("Martine␣fait␣du␣chameau␣1.png", 2048);
 ("Martine␣fait␣du␣chameau␣2.jpg", 4096); ("Photos␣2015", 6400);
 ("description␣2015.doc", 256); ("Martine␣au␣zoo␣1.jpg", 2048);
```

```
("Martine au zoo 2.jpg", 2048); ("Martine au zoo 3.png", 2048);
("Photos 2016", 8704); ("description 2016.doc", 512);
("Martine mange une pomme 1.jpg", 2048);
("Martine mange une pomme 2.jpg", 2048);
("Martine mange une pomme 3.png", 2048);
("Martine mange une pomme 4.png", 2048)]
```