Lecture et écriture de fichiers

Afin d'étendre les interactions entre nos programmes et l'utilisateur, nous allons voir comment lire et écrire des données dans des fichiers de type texte.
En Caml, il existe deux types de données prédéfinis respectivement pour les canaux d'entrée et de sortie : `in_channel` et `out_channel`.
Pour ouvrir un fichier en lecture on utilise

```
open_in : string -> in_channel!
```

`(open_in s)` ouvre le fichier de nom s en lecture et retourne le canal correspondant. L'exception `Sys_error` est levée si le fichier n'existe pas, cette exception a un argument de type string contenant un message d'erreur précis.

Pour ouvrir un fichier en écriture, on utilise

```
open_out : string -> out_channel
```

`(open_out s)` ouvre le fichier de nom s en écriture, et retourne le canal correspondant. Si le fichier existe déjà, il est préalablement effacé.
Pour fermer un fichier, on utilise

```
close_in : in_channel -> unit
close_out : out_channel -> unit
```

qui ferment les canaux donnés en argument.

La lecture depuis un fichier fonctionne de manière analogue à l'entrée au clavier :

```
input_line : in_channel -> string
```

`(input_line f)` retourne une chaîne formé d'une ligne lue depuis le canal f.
Cette procédure lève l'exception `End_of_file` si le fichier f est vide.

L'écriture dans un fichier se fait avec la fonction

```
output_string : out_channel -> string -> unit
```

`(output_string f s)` écrit la chaîne s dans le canal f.

Exercice :

Écrire une fonction qui prend en argument deux noms de fichier f_1 et f_2, lit le fichier f_1 qui devra contenir un nombre entier par ligne, et écrit le fichier f_2 qui contiendra sur chaque ligne les valeurs lues suivies de leur factorielle.
Gérer proprement les cas suivant d'erreurs :

- Le fichier f_1 est inexistant ;
- $f_1 = f_2$;
- Une ligne de f_1 ne contient pas de nombre ;
- Un de ces nombres est négatif.

Par exemple si f_1 contient

```
1
2
3
4
5
-1
x
```

l faudra fournir en sortie le fichier f_2 :

```
1 1
2 2
3 6
4 24
5 120
-1 nombre negatif
x n'est pas un nombre entier
```

La lecture et l'écriture peuvent aussi se faire caractère par caractère avec les fonctions suivantes :

```
output_char : out_channel -> char -> unit
```

`(output_char ch c)` écrit le caractère c dans le canal ch.

```
input_char : in_channel -> char
```

`(input_char ch)` lit un caractère depuis le canal ch.

Exercice :

Ecrire un programme de copie de fichier. Gérer les cas où le fichier source n'existe pas et le cas où les fichiers source et destination sont égaux.