Exercice 1 : Transformations

Q 1. Écrivez un script, nommé `csv2html`, qui convertit des lignes, lues sur l'entrée standard, en un tableau (élément `<table>`) sous la forme d'un fragment de code HTML. Chacune des lignes lues a le même format : une suite de champ séparés les uns des autres par le caractère `;`. La première ligne contient les entêtes de colonne. Pour faire vos tests, vous pouvez utiliser le fichier `bieres.csv`.

Q 2. Modifiez le script, `csv2html`, pour que

- s'il reçoit l'option `-d` suivi d'un caractère, il considère que le séparateur de champs est ce caractère ;
- s'il reçoit l'option `-s` suivi d'un nombre, le tableau doit être trié sur la colonne spécifiée par sa position ;
- s'il reçoit l'option `-S` suivi d'une chaîne, alors le tableau doit être trié sur la colonne dont le titre est spécifié.

Exercice 2 : Remplacement dans un modèle

Q 1. Écrivez un script, nommé `remplace-dans`, qui doit permettre d'afficher le contenu d'un fichier (le « *modèle* ») dont une partie est remplacée par le contenu d'un autre fichier (le « *corps* »). Le modèle est un fichier dont le nom est passé en paramètre. Le corps est lu sur l'entrée standard.

On veut obtenir un comportement semblable à celui-ci :

```
$ cat modele.txt
une ligne
--DEBUT_REMPLACEMENT--
une autre ligne
suivi encore d'une autre
--FIN_REMPLACEMENT--
la dernière ligne

$ cat corps.txt
le *corps* du "fichier" final

$ remplace-dans modele.txt < corps.txt
une ligne
le *corps* du "fichier" final
la dernière ligne
```

Q 2. Écrivez un script, nommé `remplace-dans-wiki`, qui fait la même chose que le script `remplace-dans`, mais qui en plus modifie le résultat final en appliquant certaines modifications :

- Les chaînes entourées par `*` sont modifiées pour être entourées par `<strong>` (au début) et `</strong>` (à la fin)
- Les chaînes entourées par `"` sont modifiées pour être entourées par `<em>` (au début) et `</em>` (à la fin)

Exercice 3 : Explorons le web

La commande `nc` [1] permet de rendre transparent l'échange d'un flux de caractères via le réseau.

Par exemple, la commande `nc -l 8080` affiche tout ce qui est envoyé via le réseau sur le port 8080 de la machine locale.

De la même manière `echo toto | nc 127.0.0.1 8080` envoie le mot `toto` au service écoutant sur le port `8080` de la machine locale (d'adresse `127.0.0.1`).

Q 1. Écrivez un script, nommé `plume`, qui se comporte comme un serveur web respectant le protocole HTTP version 1.0 (cf https://tools.ietf.org/html/rfc1945).

Ce serveur :

- répond toujours avec le code HTTP `200` (`OK`)
- renvoie toujours le même contenu : la chaîne `reçu`

Le script :

- ne doit traiter que les requêtes `GET`, les autres requêtes doivent provoquer un message d'erreur sur la console ;
- ne doit s'arrêter que quand la ressource demandée est nommée `exit` ;
- affiche le nom de la resource demandée sur la console.

La commande `curl` pourra sans doute pas mal vous aider dans vos tests.

1. Dans sa version écrite pour OpenBSD « *aka* » `nc.openbsd`