TP OCaml impératif

Dans ce TP d'initiation, les variables sont mutables.

Exercice 1.

1. Écrire la fonction `min (tab:'a array):'a` qui calcule le min d'un tableau ;
2. Écrire la fonction `inv (tab:'a array):'a tab` qui inverse l'ordre des éléments du tableau ;
3. Écrire `is_sorted (tab;'a array) : bool` qui indique si un tableau est trié par ordre croissant.

   On demande deux versions :
   - Dans la première, on gère une boucle `while` un compteur et un bouléen ;
   - Dans la seconde on utilise une boucle `for` et un traitement d'exception.
4. Recherche dichotomique dans un tableau trié croissant.

   Ecrire la fonction `dichot (tab:'a array) (x:'a):bool`.

   Pour contraindre le code, on utilisera deux indices :
   - un indice gauche : le premier élément à partir duquel chercher ;
   - un indice droit : le premier élément à partir duquel NE PAS chercher ;
5. Tri par insertion croissant.

   Pour cet algorithme, on maintient l'invariant suivant : le tableau est trié jusqu'à l'indice `i`. Pour traiter l'élément courant en position `i+1`, on l'insère à sa juste place dans la gauche du tableau (entre les positions 0 et `i+1`). Et on itère.

   Faire en sorte que le tri soit stable et en place.

   Ecrire `tri_insertion tab` qui trie un tableau par insertion.
6. Écrire la fonction `add (m1:int array array) (m2:int array array):int array array` qui ajoute deux matrices de dimensions compatibles.
7. Écrire la fonction `mult (m1:int array array) (m2:int array array):int array array` qui multiplie deux matrices de dimensions compatibles.

On se donne pour les exercices suivants un fichier **essai** dont voici le contenu obtenu par `cat` :

```
$ cat essai
toto
gogo

toto et gogo
```

Il nous servira pour nos tests.

Exercice 2. On attend le même comportement que `wc` de Linux.

```
$ wc essai
 3 5 23 essai
```

Cette commande compte le nombre de lignes (c.a.d le nombre de `\n`), de mots et de caractère.

1. Écrire la fonction `nbchar (nom:string) : int` qui compte le nombre de caractères dans un fichier dont le nom est donné.
2. Écrire la fonction `nbline (nom:string) : int` qui compte le nombre de lignes dans un fichier dont le nom est donné.
   Remarque. Attention, comme `input_line` supprime le `\n` de fin de ligne, si la dernière ligne est une ligne vide, elle ne sera pas comptabilisée. Notre fonction compte le nombre de lignes visibles et on tolère qu'il y ait une différence de 1 avec la réponse fournie par `wc`.
   La même tolérance s'applique aussi à la fonction précédente.
3. Écrire la fonction `nbword (nom:string) : int` qui compte le nombre de mots dans un fichier dont le nom est donné. Un *mot* est, pour simplifier, toute séquence de caractères ne comportant pas de caractères de tabulation, de passage à la ligne ou d'espace.

Exercice 3. Dans un fichier **inverser_file.ml**

1. Écrire la fonction `inverse (entree:string) (sortie:string)` qui inverse l'ordre des lignes du fichier d'entrée et écrit le résultat dans le fichier de sortie.
2. Faire en sorte pour qu'après compilation, le programme obtenu attende deux noms de fichiers et mette les lignes du premier en ordre inverse dans le second.
   Le programme affiche en outre son propre nom et les valeurs de ses arguments.

```
$ ocamlc -o inverser_file inverser_file.ml
$ ls iasse # iasse est absent
ls: impossible d'accéder à'iasse': Aucun fichier ou dossier de ce type
$ ./inverser_file essai iasse
Programme ./inverser_file :
argument 1 : essai
argument 2 : iasse

Contenu du fichier iasse:
toto et gogo

gogo
toto
$ ls iasse # iasse a été créé
iasse
$ cat iasse
toto et gogo

gogo
toto
```