Strings

Florian Bourse

Affichage

Question 1. Écrire les fonctions `mult`, `carre`, `pyramide` et `damier` permettant d'afficher les figures suivantes :

```
print_endline "mult :\n";;
print_endline (mult "O" 5);;
print_endline "carre :\n";;
print_endline (carre "O" 5);;
print_endline "pyramide :\n";;
print_endline (pyramide "O" 5);;
print_endline "damier (n impair) :\n";;
print_endline (damier "O" "X" 5);;
print_endline "damier (n pair) :\n";;
print_endline (damier "#" " " 6);;
```

```
mult :
OOOOO
carre :
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
pyramide :
O
OO
OOO
OOOO
OOOOO
damier (n impair) :
OXOXO
XOXOX
OXOXO
XOXOX
OXOXO
damier (n pair) :
# # #
 # # #
# # #
 # # #
# # #
 # # #
```

Analyse de chaînes de caractères

Question 2. Écrire une fonction `est_palindrome : string -> bool` qui prend en entrée une chaîne de caractères et qui renvoie `true` si et seulement si c'est un palindrome. *Bonus : la fonction n'accède qu'une seule fois à chaque caractère de la chaîne.*

Question 3. Écrire une fonction `est_carre : string -> bool` qui prend en entrée une chaîne de caractères s et qui renvoie `true` si et seulement si c'est un carré, c'est-à-dire qu'il existe une chaîne de caractère s' telle que $s = s'^\frown s'$.

Question 4. Écrire une fonction `check_parentheses : string -> bool` qui prend en entrée une chaîne de caractère et qui vérifie si elle est bien parenthésée. C'est-à-dire que toute parenthèse fermante rencontrée correspond à une parenthèse qui a précédemment été ouverte, et aucune parenthèse ne reste ouverte sans être fermée.

Question 5.

```
val mostly_lowercase : string -> bool = <fun>
# mostly_lowercase "azeAZEAZ";;
- : bool = false
# mostly_lowercase "azeAZEAZazeaeza";;
- : bool = true
```

Question 6. Écrire une fonction `sont_anagrammes : string -> string -> bool` qui prend en entrée deux chaînes de caractères et qui détermine si elles comportent les mêmes caractères.

Question 7. Écrire un programme qui permet de jouer au mastermind :

1. le programme choisit une chaîne de caractères ;
2. à chaque étape, l'utilisateur entre une chaîne de caractère ;
3. le programme répond combien de caractères sont présents, mais mal placés et combien son présents à la bonne position.

Références et tableaux en OCaml

Exemple d'utilisation du type ref — REPL

```
# let x = ref 0;;
val x : int ref = {contents = 0}
# x := 1;;
- : unit = ()
# x;;
- : int ref = {contents = 1}
# !x;;
- : int = 1
# x + 2;;
Line 1, characters 0-1:
1 | x + 2;;
    ^
Error: This expression has type int ref
       but an expression was expected of type int
# !x + 2;;
- : int = 3
# x := !x + 1;;
- : unit = ()
# !x;;
- : int = 2
# let y = ref false;;
val y : bool ref = {contents = false}
# !y;;
- : bool = false
# y := true;;
- : unit = ()
# !y;;
- : bool = true
```

Exemple d'utilisation du type array — REPL

```
# let t = [|0;1;2;3|];;
val t : int array = [|0; 1; 2; 3|]
# Array.length t;;
- : int = 4
# t.(1);;
- : int = 1
# t.(1) <- 4;;
- : unit = ()
# t;;
- : int array = [|0; 4; 2; 3|]
# t.(4);;
Exception: Invalid_argument "index out of bounds".
# Array.make 10 false;;
- : bool array =
[|false; false; false; false; false; false; false; false; false; false|]
# Array.init 10 (fun i -> i mod 4);;
- : int array = [|0; 1; 2; 3; 0; 1; 2; 3; 0; 1|]
# Array.make_matrix 3 2 4;;
- : int array array = [|[|4; 4|]; [|4; 4|]; [|4; 4|]|]
```