Colle 05

Cette cinquième colle vous fera reconnaître du code OCaml, et écrire quelques fonctions simples en OCaml.

Une même fonction sur une liste et un tableau

1. Écrivez en OCaml une fonction `troisieme_element_liste` qui prend en entrée une liste (de type `'a list`) et renvoie son troisième élément. On suppose que la liste est assez grande. On s'imposera de ne pas utiliser les fonctions du module `List` mais uniquement du *pattern matching* ou de la déconstruction de type (par exemple `let hd :: tl = liste in` qui déconstruit la liste en sa tête `hd` et sa queue `tl`) ;
2. De même, écrivez en OCaml une fonction `troisieme_element_tableau` qui prend en entrée un tableau (de type `'a array`) et renvoie son troisième élément. On suppose que le tableau est assez grand. On a le droit d'utiliser toutes les fonctions du module `Array`.
3. Donnez à l'écrit leur signature et expliquez.

Exponentiation et exponentiation rapide

On cherche à écrire des fonctions qui calculent l'exponentiation, c'est à dire une fonction `puissance x n` qui calcule x^n. On prendra x de type `float` et n de type `int`.

1. Écrivez une fonction impérative (c'est-à-dire avec une boucle for et une référence qui va être modifiée progressivement) qui calcule cette puissance x^n en prenant comme argument x et n (sous forme curryfiée) ;
2. Écrivez une première fonction récursive qui utilise le cas de base $x^0 = 1$ et la relation de récurrence $x^n = x * x^{n-1}$. Est-ce que cette fonction sera plus ou moins efficace que la précédente ? Quel sera son inconvénient ?
3. Écrivez une deuxième fonction récursive qui utilise le même cas de base $x^0 = 1$ mais la relation de récurrence plus complexe qui est $x^n = (x^2)^{n/2}$ si n est pair et $x^n = x * (x^2)^{(n-1)/2}$ si n est impair. On rappelle que l'on peut tester le fait que n soit pair avec $n \bmod 2 = 0$. Est-ce que cette fonction sera plus ou moins efficace que la précédente ? Quel sera son avantage ?

QCM lecture et modification de codes

Le(s)quel(s) des codes suivants sera une fonction OCaml valide ? (il faudra pouvoir justifier) Corrigez les morceaux qui ne vont pas pour les lignes incorrectes, et expliquez ce que font chaque fonction.

1. `det minimum x y = If x >= y : y else : x;;`
2. `lef maximum x = function y -> if x >= y then x else y;;`
3. `let est_plus_petit (x, y, z) = ( < ) x (maximum y z);;`
4. `def est_plus_petit = if x < y then true else false;;`
5. `lef afficher_liste_entier liste = List.iter (fun x -> print_int x; print_string " -> ") liste;;`

