Interrogation 0 du Lundi 1 Septembre

Dans l'ensemble du devoir, toutes les fonctions seront :

- *écrites en langage OCaml,*
- *précédées d'une explication des variables utilisées,*
- *précédées d'une explication de l'algorithme.*

Exercice 1 (La syntaxe OCaml)

Donner la réponse renvoyée par OCaml pour les codes suivants sous la forme `-: type = valeur` :

(a) `1 <> 0 ;;`
(b) `8/3 ;;`
(c) `[ ] ;;`
(d) `2 > 3. ;;`
(e) `[1,2;3,0] ;;`
(f) `"clemenceau".[3] ;;`
(g) `List.tl [3] ;;`
(h) `List.length ;;`
(i) `List.hd [2,3.] ;;`
(j) `[|1;2;3|].(1) <- 4 ;;`
(k) `7::3::2::[ ] ;;`
(l) `[| |]::[ ] ;;`

Exercice 2 (Itératif, récursif, récursif terminal)

1. Considérons la suite réelle (u_n) définie par :

$$\begin{cases} u_0 = 2\,, \\ \forall n \in \mathbb{N}\,,\ u_{n+1} = \sqrt{1+u_n} \end{cases}$$

   (a) Écrire en OCaml une fonction itérative qui renvoie, pour tout entier naturel `n`, la valeur de u_n.
   (b) Écrire en OCaml une fonction récursive non terminale qui renvoie, pour tout entier naturel `n`, la valeur de u_n.
   (c) Écrire en OCaml une fonction récursive terminale qui renvoie, pour tout entier naturel `n`, la valeur de u_n.

2. (a) Écrire en OCaml une fonction itérative `appartenance_iter : 'a array -> 'a -> bool` qui détermine si un élément appartient à un tableau.
   (b) Écrire en OCaml une fonction récursive `appartenance_rec : 'a list -> 'a -> bool` qui détermine si un élément appartient à une liste.

Exercice 3 (Les tris)

1. **Tri par insertion :**

   Considérons une série de n données, n appartenant à $\mathbb{N}^*$. Nous considérons en quelque sorte que les éléments à trier nous sont donnés un par un. Le premier élément à lui seul est déjà trié. Nous rangeons alors à sa bonne place le deuxième élément pour former une série triée et nous continuons ainsi, c'est-à-dire que nous insérons les éléments successivement dans une série déjà triée.

   Nous allons implémenter le tri par insertion sur les tableaux.

(a) Écrire une fonction en OCaml `insere i t` qui, le tableau t vérifiant

$$t.(0) \leqslant t.(1) \leqslant \ldots \leqslant t.(i-1),$$

le modifie de telle façon qu'il vérifie :

$$t.(0) \leqslant t.(1) \leqslant \ldots \leqslant t.(i-1) \leqslant t.(i).$$

(b) En déduire une fonction en OCaml `tri_insertion t` qui trie le tableau t d'après l'algorithme du tri par insertion.

(c) Déterminer la complexité temporelle du tri par insertion sur les tableaux.

2. **Tri fusion :**

Considérons une série de n données, n appartenant à $\mathbb{N}^*$. Le tri fusion peut se décrire avec le paradigme "diviser pour régner" de la manière suivante :

— Partition : Nous divisons la série des n données à trier en deux séries de $n/2$ données.

— Tri récursif : Nous trions ensuite les deux séries récursivement.

— Fusion : Nous terminons en fusionnant les deux séries triées.

Nous allons implémenter le tri fusion sur les listes.

(a) Écrire une fonction en OCaml `partition l` qui sépare une liste l en deux listes approximativement de même longueur suivant un schéma récursif. L'idée peut s'illustrer de la manière suivante : vous avez un paquet de cartes en main que vous distribuez alternativement à deux personnes.

(b) Écrire une fonction en OCaml `fusion l1 l2` qui fusionne deux listes triées l_1 et l_2, non nécessairement de même longueur, en une seule liste triée suivant un schéma récursif.

(c) Écrire une fonction en OCaml `tri_fusion l` qui trie une liste l d'après l'algorithme du tri fusion.

(d) Déterminer la complexité temporelle du tri fusion sur les listes.

Exercice 4 (Manipulation de fractions)

Le but de cet exercice est d'écrire une petite bibliothèque permettant de manipuler des fractions. Une fraction $\frac{a}{b}$ est représentée par deux entiers, le numérateur a et le dénominateur b. On souhaite de plus avoir les propriétés suivantes :

— la fraction est irréductible, c'est-à-dire que le PGCD de a et b vaut 1 ;

— b est toujours positif, autrement dit, le signe de la fraction est donné par le signe de l'entier a.

1. Écrire une fonction récursive `gcd : int -> int -> int` qui calcule le PGCD de deux entiers naturels a et b, en utilisant les propriétés suivantes :

$$\begin{cases} a \wedge 0 &= a \\ a \wedge b &= b \wedge a \\ a \wedge b &= (a-b) \wedge b \end{cases}$$

La fonction est-elle récursive terminale ?

2. Définir un type enregistrement `frac` possédant deux champs entiers `num` et `denom`.
3. Définir une fonction `simp f` qui simplifie la fraction f et s'assure que b est positif.
4. Définir les fonctions suivantes :
   (a) `add_frac` : additionne deux fractions ;
   (b) `neg_frac` : renvoie l'opposé ;
   (c) `sub_frac` : soustrait deux fractions ;
   (d) `mul_frac` : multiplie deux fractions ;
   (e) `inv_frac` : renvoie l'inverse ;
   (f) `div_frac` : divise deux fractions.

   Toutes ces fonctions devront renvoyer des fractions simplifiées.
