# Interrogation écrite 4

INF 201 — IMA3&4— 24/04/2026 — 20 minutes

**Exercice 1. (  /10)** On considère une fonction `fold_left` qui a le profil suivant:

```
('acc -> 'a -> 'acc) -> 'acc -> 'a list -> 'acc = <fun>
```

**Question 1. (  /2)** Quel sera le résultat de l'expression suivante:

```
List.fold_left (@) [] [[1;2;3];[4;5;6]]
```

**Question 2. (  /2)** Quel sera le résultat de l'expression suivante:

```
List.fold_left (fun x y -> if x > y then y else x) 0 [7;-6;9];;
```

**Question 3. (  /2)** On considère les déclarations suivantes:

```
let id = (fun x -> x);;
let g = List.fold_left (fun c f -> fun x -> f ( c x)) id [(fun x -> 2*x);(fun x -> x + 1)];;
```

Quel est le type de g ?

Que vaut `g 4;;` ?

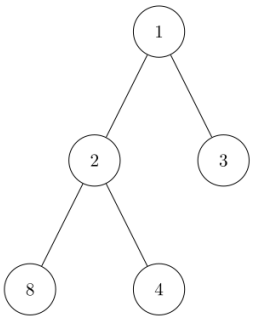
**Question 4. (  /4)** Implémenter récursivement `fold_left`.

**Question 5. (  /4)** Implémenter non récursivement une fonction `val_cum` qui retourne la liste des valeurs cumulées d'une liste d'entiers. On utilisera le schéma d'ordre supérieur `fold_left`, la fonction `List.rev` ainsi que les sélecteurs `List.hd` et `List.tl`.

**Exercice 2. (  /6)** On rappelle une définition pour le type arbre:

```
type 'a arbre = Vide | Noeud of 'a arbre * 'a * 'a arbre;;
```

**Question 6. (  /2)** Donner une implémentation dans le type `arbre` de l'arbre dessiné ci-après:




**Question 7. (  /4)** Donner une implémentation de la fonction `map_arbre` qui implémenté `map` mais pour un arbre !

```
val map_arbre : ('a -> 'b) -> 'a arbre -> 'b arbre = <fun>
```