heig-vd

# Laboratoire

## Récursivité

heig-vd

# Rapport

Attention de laboratoire fait l'objet d'un rendu noté:

- Rendre vos réponses sous la forme d'un unique module Haskell.
- Inclure votre nom et prénom et la date du rendu en commentaire au début de votre fichier.
- L'évaluation se base sur le test des fonctions exportées par le module, une fonction qui ne rend pas le résultat attendu ou qui ne compile pas ne rapporte aucun point.
- Respectez les noms de fonction demandées.
- Délai pour le rendu : prochain cours.

heig-vd

# Exercice 3.1 - Récursivité - Nombres

1. Ecrire une fonction `ack (m,n)` qui calcule la fonction d'Ackermann (2 pts) :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

2. Définir la fonction `quotient n m` qui calcule le quotient entier de n par m sans faire appel a un opérateur de division (1 pt).

3. Ecrire une fonction récursive `nieme liste n` qui retourne le nième élément d'une liste (on ne se préoccupe pas des conditions d'erreur) (1 pt).

4. Ecrire une fonction `puissance x i` récursive qui calcule $x^i$ pour i>=0 (1 pt).

heig-vd

# Exercice 3.2 - Récursivité - Listes

1. Écrire une fonction `duplique liste` qui duplique tous les éléments d'une liste (2 pts) :

   $[a_1, a_2 \ldots a_n] \rightarrow [a_1, a_1, a_2, a_2 \ldots a_n, a_n]$

2. Ecrire une fonction `renverse liste` qui renverse les éléments d'une liste (2 pts) :

   $[a_1, a_2 \ldots a_n] \rightarrow [a_n \ldots a_2, a_1]$

3. Écrire une fonction `rotation liste i` qui réalise i rotations cycliques vers la gauche :

   $[a_1, a_2 \ldots a_n] \rightarrow [a_{i+1}, a_{i+2} \ldots a_n, a_1, a_2 \ldots a_i]$

   - Il y a une solution, triviale, avec des fonctions prédéfinies, donnez la (1 pt).
   - Ecrire une solution récursive, `rotation'` (2 pts).

heig-vd

# Exercice 3.3 - Récursivité

1. On utilise une liste pour modéliser un ensemble. Ecrire les fonctions suivantes de manipulation des ensembles:

    1. `insertion liste x:`

       l'insertion dans un ensemble ajoute un élément qui n'existe pas encore (2 pts).

    2. `suppression liste x:`

       supprime un élément de l'ensemble (2 pts).

    3. `union:`

       union de deux ensembles, sans créer de doublons bien entendu (2 pts).

heig-vd

# Exercice 3.4 - Tri Fusion

1. Écrire une fonction qui `partage` une liste en deux parties égales (à un élément près) (3 pts) :

   `partage [a1,a2 ... an] → ([a1,a3 ...],[a2,a4 ...])`

   - Réaliser ce partage en une passe bien sûr !

2. Écrire une fonction qui réalise la fusion de 2 listes triées (2 pts) :

   `fusion [1,2,4] [2,5] → [1,2,2,4,5]`

3. Ecrire une fonction qui réalise le tri fusion (2 pts) :

   - Séparer la liste à trier en deux.
   - Trier puis fusionner ces deux listes.