Exercices Haskell (Partie 2)

Voici quelques exercices supplémentaires pour mettre en pratique vos connaissances en Haskell. Pour chaque exercice, créez une fonction qui effectue la tâche décrite.

1. Crée une fonction `myElem` qui prend un élément et une liste, et retourne un booléen indiquant si cet élément fait partie de la liste.

2. Crée une fonction `safeDiv` qui prend deux entiers et retourne un `Maybe Int`. Si le diviseur est égal à zéro, la fonction retourne `Nothing`, sinon elle retourne le résultat de la division dans un `Just`.

3. Crée une fonction `safeNth` qui prend une liste et un entier, et retourne l'élément à l'index spécifié dans un `Maybe`. La fonction doit retourner `Nothing` si l'index est hors de portée ou négatif.

4. Crée une fonction `safeSucc` qui prend un `Maybe Int` et retourne le successeur de l'entier encapsulé dans un `Maybe`. Si l'entrée est `Nothing`, la fonction retourne également `Nothing`.

5. Crée une fonction `myLookup` qui prend une clé et une liste de paires `(clé, valeur)`, et retourne la valeur associée à la clé dans un `Maybe`. Si la clé n'existe pas dans la liste, la fonction retourne `Nothing`.

6. Crée une fonction `maybeDo` qui prend une fonction de deux arguments et deux `Maybe`. La fonction applique l'opération aux valeurs encapsulées dans les `Maybe` si elles existent, et retourne le résultat dans un `Maybe`. Sinon, elle retourne `Nothing`.

7. Crée une fonction `isNum` qui prend une chaîne de caractères et retourne un booléen indiquant

si la chaîne contient uniquement des chiffres.

8. Crée une fonction `readInt` qui prend une chaîne de caractères et retourne un `Maybe Int`. La fonction retourne `Just Int` si la chaîne représente un nombre entier valide, sinon elle retourne `Nothing`.