HE VD
IG

Paradigmes et Langages de Programmation

Haute École d'Ingénierie et de Gestion du Canton de Vaud

1. Haskell / Syntaxe

2022

Exercice 1

Quel est le résultat des expressions ci-dessous ? Attention, certaines sont incorrectes, lesquelles ? Pourquoi ?

a. `'Z' < 'a'`
b. `"abc" <= "ab"`
c. `if 2 < 3 then 3`
d. `[1,3,3,6] < [1,3,4]`
e. `4. + 3.5`
f. `5.0 - 4.2 / 2.1`
g. `if 4 > 4.5 then 3.0 else 'a'`
h. `3 > 4 || 5 < 6 && not (7 /= 8)`
i. `if 6 < 10 then 6.0 else 10.0`

Exercice 2

Donnez les résultats des expressions ci-dessous. Attention, certaines sont incorrectes, lesquelles ? Pourquoi ?

a. `[1,2,3] !! ([1,2,3] !! 1)`
b. `head []`
c. `tail [1,2,3]`
d. `"a":["b","c"]`
e. `"abc" ++ "d"`
f. `tail "abc" ++ "d"`
g. `[1] : [2,3,4]`
h. `([1,2,3] !! 2 : []) ++ [3,4]`
i. `[3,2] ++ [1,2,3] !! head [1,2] : []`

Exercice 3

Définissez les listes suivantes :

a. Les nombres impairs de 1 à 20 (en extension)
b. Les nombres pairs de 1 à 20 (en compréhension)
c. Les carrés parfaits entre 1 et 100
d. Les paires (x, y) telles que x<7, y<7, x<y.
e. Améliorez la solution précédente pour ne pas répéter [1..7].
f. Les triplets (x, y, z) d'entiers inférieurs à 100 qui représentent la longueur des trois côtés d'un triangle rectangle.

Exercice 4

Définissez et testez les fonctions suivantes :

```
somme x y z -- Somme de x, y et z.
carre x     -- Calcul de x au carre.
triangles n -- Triplets de l'exercice precedent
ou p q      -- Le ou logique.
et p q      -- Le et logique.
```

Exercice 5

Utilisez les listes en compréhension pour lister la table de vérité de vos fonctions or et and (sous forme de triplets).

Exercice 6

Les fonctions *drop* et *take* permettent respectivement d'éliminer ou de retenir les n premiers éléments d'une liste. Comment utiliser ces fonctions pour supprimer le mot *Hello* au début d'une chaîne ? Tester si la chaîne commence par Hello ?

Exercice 7

Je suis allergique à l'américain, écrivez une fonction *clean s* qui supprime le mot *Hello* s'il figure au début de la chaîne *s*.

La phrase résultante manque de majuscule, pouvez vous corriger ça ? Utilisez *toUpper* du module *Data.Char* avec l'énoncé *import Data.Char*.

Exercice 8

Écrivez une fonction *tuple* qui retourne un triplet avec les carrés de n, $n + 1$, $n + 2$.

Exercice 9

On construit une liste avec des tuples contenant un nom et un âge. Écrivez une fonction qui donne l'âge en fonction du nom. Que se passe-t-il si le nom n'est pas dans la liste ?

Exercice 10

Familiariser vous avec les fonctions suivantes :

```
length -- Longueur d'une liste.
head   -- Premier element (la tete) d'une liste.
(!!)   -- Nieme element de la liste.
last   -- Dernier element de la liste.
tail   -- Queue de la liste.
init   -- Debut de la liste.
take   -- N premiers elements de la liste.
drop   -- N derniers elements de la liste.
(++)   -- Concatenation de deux listes.
```

Exercice 11

Définissez une fonction *slice* qui, à l'aide d'une expression de liste en compréhension, extrait les ième au jème éléments d'une liste :

```
Prelude> slice "Hello world!" 6 10
"world"
```

Définissez cette fonction de manière récursive.

Exercice 12

Écrivez une fonction *insert* qui insère un élément x à la position n d'une liste :

```
Prelude> insert 'd' 4 "abcef"
"abcdef"
```

Exercice 13

Une méthode de calcul des nombres premiers consiste à énumérer tous les entiers et successivement supprimer de la liste les multiples des éléments restants. C'est la méthode du crible d'Ératosthène.

Écrivez une fonction *sieve* qui, à l'aide d'une expression de liste en compréhension, supprime d'une liste *xs* tous les multiples de *n* :

```
Prelude> sieve 2 [1..9]
[1,3,5,7,9]
```

Exercice 14

A l'aide d'une expression de liste en compréhension, écrivez un prédicat qui vérifie si un nombre n positif est premier :

```
Prelude> prime 12
false
```

Indication : Définissez des alternatives pour les cas triviaux (1 et 2 ne sont pas considérés comme premiers).

Exercice 15

La conjecture de Goldbach est l'assertion mathématique que tout nombre entier pair supérieur à 2 est la somme de deux nombres premiers.

Écrivez une fonction *parts*, à l'aide d'une expression de liste en compréhension, qui liste les paires de nombres premiers dont la somme est égale à n :

```
Prelude> parts 14
[(3,11),(7,7),(11,3)]
```

Exercice 16

Écrire une fonction *add* d'addition de vecteurs, représentés par des paires (tuple de deux éléments) en utilisant des motifs.

Exercice 17

Définissez une fonction *prod* de produit scalaire de deux vecteurs dans l'espace (triplets).

Exercice 18

Récrivez les fonctions *add* et *prod* sur une liste à l'aide de motifs.

Exercice 19

Définir les fonctions *head'* et *tail'* à l'aide de motifs.

Exercice 20

Trouvez le cinquième élément d'une liste à l'aide d'un motif.

Exercice 21

Définissez la fonction *pgcd* qui calcule le plus grand commun diviseur de deux nombres. On se souvient qu'il suffit de retrancher successivement le plus petit du plus grand jusqu'à ce qu'ils soient identiques pour trouver ce pgcd.

Exercice 22

Écrivez une fonction *secToHMS* qui transforme des secondes en heures, minutes et secondes sans utiliser de division ni modulo.

Indication : ce problème n'est pas trivial, il faut retourner un résultat sous forme de triplet (h,m,s) et utiliser un let sur le reste pour exprimer le résultat final.

Exercice 23

Inventez une forme géométrique composée de formes élémentaires et définissez une fonction pour calculer sa surface avec un *where* comme pour l'exemple du cylindre donné dans les transparents du cours.

Exercice 24

Une manière triviale de trier une liste est d'appliquer la méthode de tri par insertion : on insère successivement tous les éléments de la liste dans une liste triée.

Écrivez une fonction qui trie par insertion une liste.

Bon travail !