Paradigmes et Langages de Programmation

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

2. Haskell / Système de types

2022

Exercice 1

Donnez une expression pour chacun des types suivants :

```
[[[Integer]]]
[(Integer, Char)]
(Double, [[Integer]])
((Integer, Integer), [Bool], Double)
```

Exercice 2

Inférez le type des expressions ci-dessous :

```
(* 9) 6
head [(0,"doge"),(1,"kitteh")]
head [(0 :: Integer ,"doge"),(1,"kitteh")]
if False then True else False
length [1, 2, 3, 4, 5]
(length [1, 2, 3, 4]) > (length "TACOCAT")
```

Exercice 3

Inférez le type des expressions ci-dessous :

```
3 * pi
(1.5,"3")
head "Hello " ++ "World !"
[[1,2],[]]
[('a',1),('b',2)]
```

Exercice 4

Inférez le type des fonctions ci-dessous :

```
second xs     = head (tail xs)
swap (x, y)   = (y, x)
pair x y      = (x, y)
double x      = x * 2
palindrome xs = reverse xs == xs
twice f x     = f (f x)
```

Exercice 5

Soient les fonctions suivantes :

a. *not*
b. *length*
c. *concat*
d. *head*
e. *(<)*

…et les signatures de type :

1. *_ : : [a] -> a*
2. *_ : : [[a]] -> [a]*
3. *_ : : [a] -> Int*
4. *_ : : Ord a => a -> a -> Bool*
5. *_ : : Bool -> Bool*

Faites correspondre chaque fonction à sa signature de type.

Exercice 6

1. Si le type de *f* est *a -> a -> a -> a* et le type de *x* est *Char*, alors le type de *f x* est :

a. *Char -> Char -> Char*
b. *x -> x -> x -> x*
c. *a -> a -> a*
d. *a -> a -> a -> Char*

2. Si le type de *g* est *a -> b -> c -> b*, alors le type de *g 0 'c' "woot"* est :

a. *String*
b. *Char -> String*
c. *Int*
d. *Char*

3. Si le type de *h* est *(Num a, Num b) => a -> b -> b*, alors le type de *h 1.0 2* est :

a. *Double*
b. *Integer*
c. *Integral b => b*
d. *Num b => b*

4. Si le type de *h* est *(Num a, Num b) => a -> b -> b*, alors le type de *h 1 (5.5 : : Double)* est :

a. *Integer*
b. *Fractional b => b*
c. *Double*
d. *Num b => b*

5. Si le type de *j* est *(Ord a, Eq b) => a -> b -> a*, alors le type de *j "keyboard" "has the word jackal in it"* est :

a. *[Char]*

b. *Eq b => b*
c. *b -> [Char]*
d. *b*
e. *Eq b => b -> [Char]*

6. Si le type de *j* est *(Ord a, Eq b) => a -> b -> a*, alors le type de *j "keyboard"* est :
   a. *b*
   b. *Eq b => b*
   c. *b -> [Char]*
   d. *b*
   e. *Eq b => b -> [Char]*

7. Si le type de *k* est *(Ord a, Num b) => a -> b -> a*, alors le type de *k 1 2* est :
   a. *Integer*
   b. *Int*
   c. *a*
   d. *(Num a, Ord a) => a*
   e. *Ord a => a*
   f. *Num a => a*

8. Si le type de *k* est *(Ord a, Num b) => a -> b -> a*, alors le type de *1 (2 : : Integer)* est :
   a. *(Num a, Ord a) => a*
   b. *Int*
   c. *a*
   d. *Num a => a*
   e. *Ord a => a*
   f. *Integer*

9. Si le type de *k* est *(Ord a, Num b) => a -> b -> a*, alors le type de *(1 : : Integer) 2* est :
   a. *Num a => a*
   b. *Ord a => a*
   c. *Integer*
   d. *(Num a, Ord a) => a*
   e. *a*

Exercice 7

1. Une valeur de type *[a]* est
   a. une liste de caractères alphabétiques
   b. une liste de listes
   c. une liste où les éléments sont tous de type *a*
   d. une liste où les éléments sont tous de type différent

2. Une fonction de type *[[a]] -> [a]* peut
   a. prendre une liste de chaînes de caractères en argument

b. transformer un caractère en chaîne de caractères
c. transformer une chaîne de caractères en une liste de chaînes de caractères
d. prendre deux arguments

3. Une fonction de type *[a] -> Int -> a*

a. prend un seul argument
b. retourne un élément de type *a* de la liste
c. doit retourner une valeur de type *Int*
d. est complètement fictive

4. Une fonction de type *(a, b) -> a*

a. prend une liste en argument et retourne une valeur de type *Char*
b. a zéro arguments
c. prend un tuple en argument et retourne la première valeur
d. exige que *a* et *b* soient de types différents

Exercice 8

1. Sachant

```
x = 5
y = x + 5
w = y * 10
```

Quel est le type de *w* ?

2. Sachant

```
x = 5
y = x + 5
z y = y * 10
```

Quel est le type de *z* ?

3. Sachant

```
x = 5
y = x + 5
f = 4 / y
```

Quel est le type de *f* ?

4. Sachant

```
x = "Julie"
y = " <3 "
z = "Haskell"
f = x ++ y ++ z
```

Quel est le type de *f* ?

5. Sachant

```
x = 3.14
y = x + 5
z = show y
```

Quel est le type de *z* ?

Exercice 9

On souhaite modéliser des ensembles à travers des listes Haskell. Décrivez les signatures des fonctions de manipulation d'ensembles suivantes :

- Insertion d'un élément x
- Suppression d'un élément x
- Appartenance d'un élément x
- Union de deux ensembles s_1 et s_2
- Différence de deux ensembles s_1 et s_2

Implémentez ces fonctions.

Exercice 10

Soit l'extrait de code suivant :

```
d (v, e) n
  | null [] = []
  | otherwise = d' (v,e) [n]

d' ([],_) _ = []
d' (_,_) [] = []
d' (v,e) (t:r)
  | [x | x <-v , x == t] == [] = d' (n, e) r
  | otherwise = t : d' (n, e) (a ++ r)
  where
    a = [x | (x,y) <- e, y== t ] ++ [ x | (y,x) <- e, y == t]
    n = [x | x <- v, x /=t ]
```

Inférez le type des fonctions *d* et *d'*.

Bon travail !