Exercice 1 : Expressions 1

Pour chacune des expressions suivantes, donner, si elle est correcte, son type et sa valeur.

1.
```
let (a, b) = (2, 3) in
(a + b, b)
```

2.
```
let a, b = true, false in
if b then 1 else 3.5
```

3.
```
(fun g x -> g x)
  ((fun a x -> x + a) 1)
  (1)
```

Exercice 2 : Expressions 2

Pour chacune des expressions suivantes, donner, si elle est correcte, son type et sa valeur.

1.
```
let y = (5, "true") in
let (a, b) = y in
b
```

2.
```
let a, b = true, false in
if b then 1 else if a then 3 else 2
```

3.
```
let a =
  (fun b ->
     if b then fun x -> (3 * x)
     else fun x -> (2 * x))
in
(a true 1) * (a false 1)
```

Exercice 3 : Expressions 3

Pour chacune des expressions suivantes, donner, si elle est correcte, son type et sa valeur.

1.
```
let x = ('a', 'b') in
let (a, b) = x in
a
```

2.
```
let a, b = true, "" in
if b then 1 else 2
```

3.
```
let a = (fun b ->
  if b then fun x -> (x+1)
  else fun x -> (x+2))
in
let t = (fun x -> x mod 2 = 0) in
(a (t 0) 0) + (a (t 1) 1)
```

Exercice 4 : Heures

On représente un *moment* dans la journée, par une paire d'entiers dénotant les heures et les minutes.

Q. 1 Définir une fonction OCaml `heure : int -> int -> (int * int)` permettant de fabriquer un moment à partir d'un nombre d'heures et d'un nombre de minutes.

Q. 2 Définir une fonction OCaml `apres : (int * int) -> (int * int) -> bool` permettant de tester si un moment m_1 si situe plus tard dans la journée qu'un moment m_2 au moyen de l'appel `apres` m_1 m_2.

Q. 3 Définir une fonction OCaml permettant de calculer le temps qui s'est écoulé, en minutes entre deux moments `ecoule : (int * int) -> (int * int) -> int`

Q. 4 Définir une fonction OCaml permettant de calculer si un moment x se trouve entre deux moments m_1, m_2.

Exercice 5 : Intervalles d'entiers

On représente un intervalle d'entier $[\![a, b]\!]$ en OCaml par une pair `(a, b)`.

Q. 1 Définir une fonction OCaml `singleton : int -> (int * int)` prenant en argument un entier et calculant l'intervalle contenant uniquement cet entier.

Q. 2 Définir une fonction OCaml `card: (int * int) -> int` prenant en argument un intervalle d'entiers et retournant son nombre d'éléments.

Q. 3 Définir une fonction OCaml `mem : int -> (int * int) -> bool` prenant un argument un entier x et un intervalle d'entier I et calculant si oui ou non $x \in I$.

Q. 4 Définir une fonction OCaml `intersection : (int * int) -> (int * int) -> (int * int)` prenant en argument deux intervalles et calculant leur intersection.

Exercice 6 : Multiplication

Dans cet exercice on ne s'autorise pas la multiplication d'entiers en OCaml

Q. 1 Proposer une mise en équation récursive de l'opération $a \times b$ où a et b sont des entiers naturels.

Q. 2 Proposer une fonction OCaml calculant $a \times b$ à partir des entiers a et b.

Exercice 7 : Addition

Dans cet exercice on ne s'autorise pas l'addition d'entiers en OCaml, seulement l'incrémentation/la décrémentation par 1.

Q. 1 Proposer une mise en équation récursive de l'opération $a+b$ où a et b sont des entiers naturels.

Q. 2 Proposer une fonction OCaml calculant $a + b$ à partir des entiers a et b.

Exercice 8 : Nombre de chiffres dans un nombre

Q. 1 Proposer une mise en équation récursive du nombre de chiffres dans un nombre n.

Q. 2 Proposer une fonction OCaml calculant le nombre de chiffres dans un nombre n passé en argument.

Exercice 9 : Ordre supérieur

Q. 1 Un sous-ensemble X d'un ensemble S peut être vu comme une fonction de S dans l'ensemble $\{0, 1\}$. Expliciter une telle fonction.

On représente donc un ensemble d'entiers en OCaml par une fonction de type `int -> bool`.

Q. 2 Définir une fonction `singleton : int -> (int -> bool)` calculant, étant donné un entier x, l'ensemble $\{x\}$.

Q. 3 Définir une fonction `mem : int -> (int -> bool) -> bool` prenant en argument un entier x et un ensemble d'entiers X et calculant vrai si et seulement si $x \in X$.

Q. 4 Définir une fonction `complementaire : (int -> bool) -> (int -> bool)` calculant le complémentaire d'un ensemble d'entier.

Q. 5 Définir une fonction `union : (int -> bool) -> (int -> bool) -> (int -> bool)` prenant en argument deux ensembles d'entiers et calculant leur union.

Exercice 10 : Pseudo Inverse 1

Q. 1 Écrire une fonction `inv_exp2: int -> int` telle que `(inv_exp2 n)` renvoie le plus grand $d \in \mathbb{N}$ tel que $2^d \leqslant$ `n`. On supposera que `n` $\geqslant 0$.

Q. 2 Quelle est la complexité de votre fonction ?

Exercice 11 : Pseudo Inverse 2

Q. 1 Écrire une fonction `inv_square: int -> int` telle que `(inv_square n)` renvoie le plus grand $d \in \mathbb{N}$ tel que $d^2 \leqslant$ `n`. On supposera que `n` $\geqslant 0$.

Q. 2 Quelle est la complexité de votre fonction ?

Exercice 12 : Pseudo Inverse 3

Q. 1 Écrire une fonction `inv_powpow: int -> int` telle que `(inv_powpow n)` renvoie le plus grand $d \in \mathbb{N}$ tel que $2^{2^d} \leqslant$ `n`. On supposera que `n` $\geqslant 0$.

Q. 2 Quelle est la complexité de votre fonction ?

Exercice 13 : Image des triplets

Q. 1 Donner une fonction `sum_img_triple` prenant en arguments une fonction f, un entier n et calculant $\sum_{(i,j,k)\in[\![0,n-1]\!]^3} f(i,j,k)$. On précisera le type d'une telle fonction

Q. 2 En déduire une fonction prenant en argument un entier n et calculant $\sum_{i=0}^{n-1} i^3$.

Q. 3 Généraliser la fonction `sum_img_triple` pour pouvoir calculer autre chose qu'une somme des $f(i,j,k)$. Par exemple, le produit, ou la concaténation.

Q. 4 Votre fonction `sum_img_triple` est elle récursive terminale ? Si non la rendre récursive terminale.

Exercice 14 : Image des pairs

Q. 1 Donner une fonction `sum_img_triple` prenant en arguments une fonction f, un entier n et calculant $\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} f(i,j)$. On précisera le type d'une telle fonction

Q. 2 En déduire une fonction prenant en argument un entier n et calculant $\sum_{i=0}^{n-1} i^2$.

Q. 3 Généraliser la fonction `sum_img_pair` pour pouvoir calculer autre chose qu'une somme des $f(i,j)$. Par exemple, le produit, ou la concaténation.

Q. 4 Votre fonction `sum_img_pair` est elle récursive terminale ? Si non la rendre récursive terminale.

Exercice 15 : Image d'une progression géométrique

Q. 1 Donner une fonction `sum_img_prog` prenant en arguments une fonction f, un entier x, un entier n et calculant $\sum_{i=0}^{n-1} f(x^i)$. On précisera le type d'une telle fonction.

Q. 2 En déduire une fonction prenant en argument un entier n et calculant $\sum_{i=0}^{n-1} 2^{2i}$.

Q. 3 Généraliser la fonction `sum_img_prog` pour pouvoir calculer autre chose qu'une somme des $f(x)$. Par exemple, le produit, ou la concaténation.

Q. 4 Votre fonction `sum_img_prog` est elle récursive terminale ? Si non la rendre récursive terminale.

Exercice 16 : Polymorphisme 1

Q. 1 Donner une fonction `applique3` prenant en argument un triplet de fonction f, g, h et un triplet de valeurs x, y, z et retournant le triplet des images $f(x), f(y), f(z)$. Quel est le type le plus général de la fonction `applique3`.

Exercice 17 : Polymorphisme 2

Q. 1 Donner une fonction `tri2` prenant en arguments deux fonctions f et g et une valeur x et retournant la paire (f,g) si $f(x) < g(x)$ et la paire (g,f) sinon. Quel est le type le plus général de la fonction `tri2`.

Exercice 18 : Polymorphisme 3

Q. 1 Donner une fonction `comp_if` prenant en arguments trois fonctions f, g et h et un booléen b et retournant $f \circ g$ si b est vrai et $f \circ h$ sinon. Quel est le type de votre fonction ?

Exercice 19 : Entier retourné

Q. 1 Définir une fonction `fst_digit : int -> int * int` prenant en argument un entier naturel et retournant une paire telle que si `fst_digit z` $= (x, y)$ alors x est le premier chiffre de l'écriture de z en base 10 et y est le reste des chiffres de cette écriture. Par exemple `fst_digit 1234 = (1, 234)`.

Q. 2 En déduire une fonction `prepend : int -> int -> int` prenant en arguments un entier naturel x de $[\![0, 9]\!]$ et un entier y et retournant le nombre obtenu en plaçant le digit x en tête du nombre y. Par exemple `prepend 1 234 = 1234`

Q. 3 En déduire une fonction `rev: int -> int` retournant un entier naturel. Par exemple `rev 1234 = 4321`.

Q. 4 Donner une version récursive terminale de `rev`.

Exercice 20 : Entier divisible par 9

Q. 1 Écrire une fonction `sum_digits : int -> int` calculant la somme des digits d'un entier naturel.

Q. 2 Sans utiliser la fonction `mod` et sans la redéfinir écrire une fonction `div9 : int -> bool` calculant si un nombre est ou non un multiple de 9.

Exercice 21 : Puissance de 2 moins 1

Q. 1 Donner une fonction `forall` prenant en arguments une fonction f et un entier n. `forall` calcule si la fonction f, appliquée sur chacun des bits de l'écriture en base 2 de n vaut vrai. Un bit sera représenté au moyen d'un booléen `true` ou `false`. Donner le type de votre fonction `forall`.

Q. 2 En déduire une fonction `pow2m1 : int -> int` permettant de tester si un nombre est de la forme $2^p - 1$.

Exercice 22 : Entiers palindromes

Q. 1 Écrire une fonction permettant de tester si un entier est un palindrome.