Informatique
MP2I 2022/2023 — DS1 — CIV

Durée totale 2h.
Tout document interdit. Tout dispositif électronique interdit (calculatrice, cafetière, R2D2...)
On vous demande d'écrire lisiblement et de bien présenter. Les programmes en particulier doivent être bien indentés et lisibles.
Bon courage et *Don't panic!* 😃

1 Fonctions proches du TP

⚠ on utilise ici les types par défaut de `OCaml`. C'est particulièrement important pour les listes.
Quand on vous dit que $n \geq 0$, c'est acquis que n ne sera jamais négatif donc ne vous en préoccupez pas dans les questions et ne faites jamais de `if n < 0 then failwith`....

Q01 Écrire une fonction récursive `fact : int -> int` telle que `fact n` renvoie $n!$ où $n \geq 0$

Q02 Écrire une fonction `pow2 : int -> int` telle que `pow2 n` renvoie 2^n où $n \geq 0$ de manière naïve.

Q03 Écrire une fonction `pow2_rap : int -> int` telle que `pow2_rap n` renvoie 2^n où $n \geq 0$ en remarquant que :

$$2^{2p} = (2^p)^2 \qquad 2^{2p+1} = 2 \times 2^{2p}$$

Q04 Écrire une fonction récursive `suplog2 : int -> int` telle que `suplog2 n` renvoie le plus petit $d \in \mathbb{N}$ tel que $0 \leq n < 2^d$ avec la contrainte suivante :

1. en utilisant `pow2`
2. sans utiliser `pow2`

Q05 Écrire une fonction `applique : ('a -> 'b) -> 'a list -> 'b list` telle que l'appel à `applique f l` renvoie la liste obtenue en appliquant à chaque élément de `l` la fonction `f`.

Q06 Écrire une fonction `nieme : 'a list -> int -> 'a` telle que l'appel à `nieme l n` renvoie `x` où `x` est le n-ième élément de `l` et déclenche une erreur s'il n'y en a pas.

2 Fonctions et typage

Q07 Écrire une fonction `evalue` telle que `evalue f x` revoie la valeur de la fonction `f` évaluée en `x`.
Attention la fonction est vraiment très simple.

Q08 Quel est le type de la fonction précédente ?

Q09 Écrire une fonction `compose` telle que `compose f g x` calcule la valeur de $(f \circ g)(x)$ si on utilisait la notation mathématique (utiliser la notation OCaml revient ici à répondre à la question).

Q10 Quel est le type de la fonction précédente ?

3 Animal de compagnie

On considère ici le type suivant :

```
type pet = Cat | Dog
```

Q11 Écrire une fonction `ennemy` qui renvoie l'animal opposé. Par exemple, `ennemy Cat` renvoie `Dog`.

On dit que l'age, converti en années humaines, d'un chien est de 7 fois son age réel.

Q12 Écrire une fonction `dog_years` qui prend en entrée l'age réel d'un chien et renvoie son age converti.

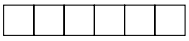
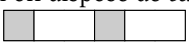
On dit que l'age, converti en années humaines, d'un chat est :
- de 12 ans si le chat a un 1 an
- de 24 ans si le chat a 2 ans
- et chaque année supplémentaire à partir de la troisième compte comme 5 années humaines

Ainsi, un chat de 4 ans a $24 + 2 \times 5 = 34$ années humaines.

Q13 Écrire une fonction `cat_years` qui prend en entrée l'age réel d'un chat et renvoie son age converti.

Q14 En déduire une fonction `pet_years : pet -> int -> int` qui calcule l'age converti d'un animal selon sa race.

4 Couvertures

On considère une ligne de n cases carrés :  que l'on souhaite recouvrir complètement avec des tuiles de différentes longueurs.
Par exemple, si on dispose de tuiles de longueur 1 ou 2, voici une couverture valide avec deux tuiles de longueur1 et deux de longueur 2. 

Q15 Écrire une fonction OCaml `couvertures12 : int -> int` telle que `couvertures12 n` calcule le nombre de couvertures de n cases avec des tuiles de longueurs 1 ou 2.
Indication : on pourra raisonner sur la tuile la plus à gauche d'une telle couverture.

Q16 En relisant la fonction précédente, reliez ce nombre de couvertures à une suite classique.

Q17 Montrer qu'il est équivalent de couvrir n cases ainsi et de couvrir un échiquier de $2 \times n$ cases avec des dominos.
Rappel : un domino est un rectangle de deux cases de long, on peut donc le placer verticalement ou horizontalement pour couvrir deux cases de l'échiquier.

Q18 Déterminer une condition sur n et m pour qu'un échiquier de $n \times m$ cases puisse être recouvert par des dominos.

Q19 **(difficile)** Écrire une fonction OCaml `couvertures : int list -> int -> int` telle que `couverture l n` calcule le nombre de couvertures de n cases avec des tuiles de longueur données par la liste l.
Par exemple, la fonction précédente correspondra à `couvertures [1;2]`
On ne suppose rien sur l. Si un nombre est présent plusieurs fois, c'est comme si on considérait des tuiles de même longueur mais de couleurs différentes, ce qui augment le nombre de couvertures.

Q20 Calculer le nombre de couvertures pour la liste `[1;1;1;1]` qui contient donc quatre fois une tuile de longueur 1 pour quatre couleurs différentes.

