OCamL all the way down
Matthieu Boyer
Planche A

Khôlle MP2I
07/10

Question de Cours

```
1 + 2;;
2. +. 3.14 /. 3.14;;
15 - 3 mod 4.;;
1. + 1. / 3.;;
17 - 3 * 6;;
```

Pendule

On définit les suites récursives suivantes :

$$\begin{cases} u_{n+1} = & 3u_n + v_n \\ v_{n+1} = & -u_n + v_n \end{cases}$$

1. Donner deux fonctions mutuellement récursives pour calculer u et v.
2. En considérant (u, v), calculer u et v en une seule fonction récursive.

Radixisation

Donner une fonction récursive permettant de renvoyer l'écriture en base b d'un entier n.

OCamL all the way down
Matthieu Boyer
Planche A : Corrigés

Khôlle MP2I
07/10

OCamL all the way down
Matthieu Boyer
Planche B

Khôlle MP2I
07/10

Question de Cours

```
1 + 4;;
(4 / 3) + 6 mod 4;;
(8 * 3) /. 4.;;
4. - (1. * 3.);;
12 - (4 / 6);;
```

Caisse Parlante

On dispose d'un stock illimité de billets et de pièces de valeurs $c_1, \ldots, c_p$. On souhaite dénombrer le nombre de manières possibles d'obtenir une somme donnée avec ces espèces. Écrire une fonction récursive prenant en paramètre la somme à atteindre et la liste des valeurs de pièces et calculant cette quantité. On pourra montrer :

$$\nu(X) = \sum_{i=1}^{p} \mathbb{1}_{c_i \leq X} \times \nu(X - c_i)$$

Tricoloration de Tableaux

Soit T un tableau de taille $2n$ dont les valeurs sont dans $\{0, 1, 2\}$. On dit que T est tricoloré si :

$$\forall 0 \leq i < N, T[i] \neq T[2i+1] \neq T[2i+2] \neq T[i]$$

1. Dénombrer le nombre de tableau tricolorés de longueur $2^k - 1$

On dit plus généralement que T de taille $(k-1)n$ est k-coloré si :

$$\forall 0 \leq i < n, \forall 0 \leq j_1 < j_2 \leq k-1, T[i+j_1] \neq T[i+j_2]$$

1. Donner une fonction permettant de vérifier si un tableau est k-coloré.
2. Donner une fonction récursive permettant de calculer une k-coloration d'un tableau. Elle prendra comme paramètres k et n et renverra une k-coloration du tableau.

Solution: Caisse Parlante

On va utiliser la relation de récurrence suivante :

$$\nu(X) = \sum_{i=1}^{p} \mathbb{1}_{c_i \leq X} \times \nu(X - c_i)$$

On propose alors le code OCamL suivant :

```
coins = [1, 2, 5, 10, 20, 50] (*On récupère les pièces sous forme de liste*)

let rec count = function
        | 0 -> 1
        | n ->
        let rec sumlist = function
                | [] -> 0
                | h::t -> h + (sumlist t)
        in sumlist (List.map (k -> if k <= n then count (n - k) else 0))
;;
```

Solution: Tricoloration de Tableaux

On montre par récurrence sur k que $N_k = 3 \times 2^{2^k - 1}$

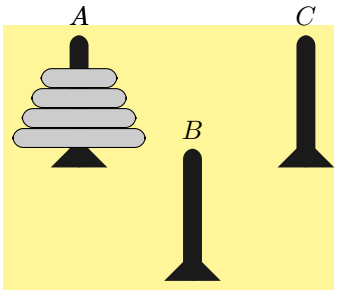
- Pour $k = 1$, il n'y a aucune condition sur l'unique case, on a donc 3 choix, et $N_1 = 3 = 3 \times 2^0 = 3 \times 2^{2^0 - 1}$
- Par récurrence, soit une configuration de la première moitié du tableau, on a maintenant la deuxième moitié du tableau à choisir. Pour chaque couple $(T[2i+1], T[2i+2])$, on a que deux choix, qui sont les deux couleurs (a puis b ou b puis a) non utilisées par $T[i]$. On trouve donc $N_{k+1} = N_k \times 2^{2^k - 2}$. En résolvant la relation de récurrence, on obtient bien le résultat.

OCamL all the way down
Matthieu Boyer
Planche C

Khôlle MP2I
07/10

Tours de Hanoï

Les *tours de Hanoï* est est un puzzle inventé par le mathématicien français Édouard Lucas : il est constitué de trois tiges sur lesquelles peuvent être enfilés n disques de diamètres différents. Au début du jeu, ces disques sont tous enfilés sur la même tige, du plus grand au plus petit.



On ne peut déplacer qu'un disque à la fois, l'objectif étant déplacer tous les disques sur une autre des tours. Donner le code d'une fonction résolvant le jeu en imprimant tous les mouvements effectués.

Solution: Tours de Hanoï

```
using namespace std;

void towerOfHanoi(int n, char from_rod, char to_rod,
        char aux_rod)
{
        if (n == 0) {
                return;
        }
        towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
        cout << Move disk  << n <<  from rod  << from_rod
                <<  to rod  << to_rod << endl;
        towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
```