Itérations et Récursivité

I. Quelques exercices de base

Exercice 1 Écrire une fonction qui calcule les coefficients du binôme...

1. en utilisant la formule de Pascal,
2. puis en utilisant la relation
$$\binom{n+1}{k+1} = \frac{n+1}{k+1}\binom{n}{k}$$
.

Exercice 2 – PGCD et coefficients de Bézout

1. Écrire (ou réécrire) une fonction `pgcd` qui retourne le PGCD des deux entiers passés en argument.
2. Écrire une fonction `bezout` qui, en plus du PGCD, retourne les coefficients de Bézout.
3. (Plus dur) Pouvez-vous écrire une version récursive terminale de la fonction précédente ?

Exercice 3 – Rendu de monnaie On dispose d'un stock illimité de pièces et de monnaie de valeurs $c_1, c_2, \dots, c_n$ euros et on souhaite connaître le nombre de façons d'obtenir n euros à l'aide de ces pièces et billets.

L'objectif de l'exercice est donc d'écrire une fonction

```
rendu : int -> int list -> int
```

tel que, par exemple,

```
# rendu 50 [20; 10; 5; 2; 1] ;;
- : int = 450
```

Vous pourrez commencer par répondre aux deux questions suivantes :

- ⋆ Combien y-a-t-il de décompositions de n n'utilisant pas la pièce c_1 ?
- ⋆ Et au moins une fois ?

Exercice 4 – Partition d'un entier

Il y a 3 façons de décomposer 6 comme somme de trois entiers naturels non nuls :

$$6 = 1+1+4 \quad 6 = 1+2+3 \quad 6 = 2+2+2$$

Plus généralement, on note $p(n,k)$ le nombre de partitions de l'entier n comme somme de k entiers strictement positifs, et on définit

$$p(n) = \sum_{k=1}^{n} p(n,k)$$

1. Justifier que pour tous $1 < k < n$,
$$p(n,k) = p(n-1,k-1) + p(n-k,k)$$
2. Déterminer des « cas de base » donnant des valeurs pour les $p(n,k)$ sans faire intervenir de relation de récurrence.
3. En déduire une fonction `partitions : int -> int` qui, étant donné un entier n, renvoie $p(n)$. Pour vérification, on indique que $p(10) = 42$.

Exercice 5 – Dessins récursifs On va s'intéresser à la courbe du dragon. Avant cela, voyons quelques instructions pour tracer des figures en OCaml.

Commençons par charger la bibliothèque graphique et ouvrir une fenêtre dans laquelle dessiner.

```
#load "graphics.cma" ;;
open Graphics ;;
open_graph "" ;;
```

Nous avons maintenant à notre disposition les fonctions suivantes :

- ⋆ `size_x ()` et `size_y ()` renvoient la largeur et la hauteur de la fenêtre graphique.
- ⋆ `clear_graph ()` efface la fenêtre graphique.
- ⋆ `current_x ()` et `current_y ()` permettent d'obtenir les coordonnées actuelles du curseur.
- ⋆ `moveto x y` déplace le curseur aux coordonnées indiquées sans effectuer de tracer.
- ⋆ `lineto x y` trace un segment de droite de la position actuelle jusqu'à celle indiquée qui sera la nouvelle position du curseur.

Au début, le curseur est aux coordonnées $(0,0)$.

On définit la *courbe du Dragon* d'ordre n du point P au point Q de la façon suivante :

- ⋆ si $n = 0$, il s'agit du segment reliant P et Q ;

- ⋆ sinon, on détermine le point R tel que PRQ soit un triangle isocèle rectangle *direct* en R, et on trace successivement les courbe du Dragon d'ordre $n-1$ de P à R puis de Q à R (attention à l'ordre).

1. Écrire la fonction

   ```
   dragon : int -> int -> int -> int -> int -> unit
   ```

   telle que `dragon n x1 y1 x2 y2` trace la courbe du Dragon d'ordre `n` du point de coordonnées (x_1, y_1) au point de coordonnées (x_2, y_2).

2. Pouvez-vous écrire la fonction en n'utilisant que des `lineto` ?