Institut National Universitaire Champollion

TP 4 - Récursivité sur les nombres

Programmation fonctionnelle en Caml
L3 INFO - Semestre 6

1 - Puissance naïve

1. Écrire une fonction `puissance : int * int -> int` qui calcule la puissance n-ième (n est un entier) d'un réel x.
   Méthode : Exprimer x^n en fonction dde x^{n-1}. Le cas de base et l'appel récursif seront distingués par filtrage.
2. Combien d'appels récursifs le calcul de `puissance(3,4)` exige-t-il ? Et plus généralement de `puissance(x,n)` ?

2 - Quelques fonctions sur les nombres

Écrire les fonctions récursives suivantes :

1. `repet(c,n) : int * int -> int` qui construit le nombre obtenu en répétant n fois le chiffre c.

```
repet(5,4) ; ;
-  : int = 5555
```

2. `unChiffre(n,c) : int * int -> bool`, qui renvoie `true` si et seulement si le nombre n n'est constitué que du chiffre c.

```
unChiffre(555755,5) ; ;
-  : bool = false
unChiffre(55555,5) ; ;
-  : bool = true
```

3. `pgd(n,d)` qui retourne le plus grand nombre entier inférieur ou égal à d et divisant n. *On rappelle que* 1 *diviste tous les entiers...*

```
pgd(18,12) ; ;
-  : int = 9
pgd <-- 7, 6
-  : int = 1
```

4. `nbPairChif(n)` qui renvoie `true` si et seulement si n contient un nombre pair de chiffres.

```
#nbPairChif(456789) ; ;
-  : bool = true
#nbPairChif(4567899) ; ;
-  : bool = false
```

3 - Multiplication égyptienne

Les égyptiens du deuxième millénaire avant J.C. utilisaient une numération additionnelle de base 10 (ce que vous connaissez de plus proche est la numération romaine...). Avec ce type de numération, il est très simple d'ajouter deux nombres, donc de faire des produits par 2 et de faire des divisions entières par deux. Pour les autres produits, c'est moins simple...

Les scribes égyptiens ont mis au point une technique de multiplication récursive n'utilisant que des sommes et des partages en 2. Elle peut se décrire avec le vocabulaire d'aujourd'hui de la manière suivante :

- Si n est pair, alors $n \times p = (n/2) \times (p+p)$
- Si n est impair, alors $n \times p = (n-1) \times p + p$.

1. Écrire une fonction `mult_egypt(n,p) : int * int -> int` qui calcule $n \times p$ en exploitant cette idée récursive.
2. Décrivez *à la main* la suite des appels (récursifs) engendrés par `mult_egypt(59,17)`.
3. Vérifiez votre prédiction en traçant l'exécution de la fonction `mult_egypt`.

4 - Somme de chiffres

On souhaite écrire en Caml des fonctions calculant la somme des chiffres d'un nombre entier représenté en base 10.

1. Définir une fonction récursive `s_chif : int -> int` qui calcule la somme des chiffres d'un entier naturel.

```
s_chif (302091) ; ;
- : int = 15
```

2. Écrire une fonction `som_chif : int -> int` qui poursuit le calcul de la somme des chiffres d'un nombre jusqu'à se ramener à un seul chiffre.

```
som_chif(18925) ; ;
- : int = 7
```