Exercice - 22 - X M_2 2012 - à la mode pour 2015

On appelle ici A l'ensemble des polynômes de $\mathbb{Z}[X]$ dont tous les coefficients valent 0 ou 1. Montrer que

$$\forall n \in \mathbb{Z}, \exists\, !P \in A;\ n = P(-2)$$

Solution.
Démontrer l'existence et l'unicité de la décomposition en base -2 des entiers naturels suffira. Elle découle directement de l'existence (et de l'unicité de l'écriture) de la division euclidienne pour les entiers relatifs, à condition d'imposer au reste d'être positif.

Division Euclidienne dans $\mathbb{Z}$

Soit $(a,b) \in \mathbb{Z} \times \mathbb{Z}^*$. Il existe un unique couple (q,r) d'entiers tels que :

$$\begin{cases} a &= bq + r \\ r &\in [\![0, |b| - 1]\!] \end{cases}$$

En effet : Soit $(a,b) \in \mathbb{Z} \times \mathbb{Z}^*$.

— Existence :
Si $b|a$, c'est immédiat.

Sinon : $|a| \overset{\text{DE}}{=} |b|q_0 + r_0$ (avec $0 < r_0 < |b|$)
L'existence d'entiers q, r convenant est assurée en choisissant :

Couple d'entiers convenant		
Si	$b > 0$	$b < 0$
$a \geqslant 0$	$q \overset{\text{déf}}{=} q_0,\ r \overset{\text{déf}}{=} r_0$	$q \overset{\text{déf}}{=} -q_0,\ r \overset{\text{déf}}{=} r_0$
$a \leqslant 0$	$q \overset{\text{déf}}{=} -q_0 - 1,\ r \overset{\text{déf}}{=} b - r_0$	$q \overset{\text{déf}}{=} q_0 + 1,\ r \overset{\text{déf}}{=} -b - r_0$

— Unicité :
Si (q,r) et (q',r') conviennent : $bq + r = bq' + r'$, et $r - r' = b(q' - q)$ est un multiple de b appartenant, par hypothèse, à $[\![1 - |b|, |b| - 1]\!]$; d'où $r - r'$ est nul, et, par suite, $q' - q$ aussi.

L'existence et l'unicité de la décomposition des entiers en base -2 est donnée par :

Algorithme : Construction de la décomposition en base -2

Données : entier n
Résultat : Décomposition en base -2 de n
$n \overset{\text{DE dans } \mathbb{Z}}{=} -2q_0 + r_0$ (avec $r_0 \in \{0,1\}$) ;
$k \overset{\text{déf}}{=} 0$;
tant que $q_k \neq 0$ **faire**
  $q_k \overset{\text{DE dans } \mathbb{Z}}{=} -2q_{k+1} + r_{k+1}$;
  On a alors

$$\begin{cases} \forall i \leqslant k+1, r_i \in \{0,1\} \\ n = r_0 + (-2)^1 r_1 + \cdots + (-2)^{k+1} r_{k+1} + (-2)^{k+2} q_{k+1} \end{cases}$$

  $k \longleftarrow k + 1$;
fin
A l'issue de la boucle, on a :

$$\begin{cases} \forall i \leqslant k, r_i \in \{0,1\} \\ n = r_0 + (-2)^1 r_1 + \cdots + (-2)^k r_k \end{cases}$$

— Terminaison : La terminaison de l'algorithme est assurée par la **stricte décroissance** de la suite positive d'entiers $|q|$, qui converge nécessairement vers 0.

— CORRECTION : la correction est immédiate, par construction[1].

Pour revenir aux notations de l'énoncé :
Si n est un entier naturel, on lui applique l'algorithme précédent, et en posant $\mathrm{P} \overset{\text{déf}}{=} r_0 + r_1\mathrm{X} + \cdots + r_k\mathrm{X}^k$, P est l'unique polynôme de A tel que $n = \mathrm{P}(-2)$. □

Script Python

ENTRÉE

```
def div_eucl(a,b=-2):
    q,r = divmod(abs(a),abs(b))
    if r==0:
        return (q,0) if ((a >= 0 and b > 0) or (a <= 0 and b < 0)) else (-q,0)
    elif a >= 0 and b > 0:
        return q,r
    elif a >= 0 and b < 0:
        return -q,r
    elif a <= 0 and b > 0:
        return -q-1,b-r
    else :
        return q+1,-b-r

def Polynome_negabinaire(n):
    C = [] # Coefficients du polynome
    q,r = div_eucl(n)
    C.append(r)
    while q != 0:
        q,r = div_eucl(q)
        C.append(r)
    return C

Polynome_negabinaire(10)
```

SORTIE (iPython) :

```
In [1]: Polynome_negabinaire(19)
Out[1]: [1, 1, 1, 0, 1]
```

Script CamL

ENTRÉE

```
let div_eucl a b = let q,r = abs(a)/abs(b), abs(a) mod abs(b) in
    if r=0 then
        if ((a >= 0 && b > 0) || (a <= 0 && b < 0)) then (q,0)
        else (-q,0)
    else if (a >= 0 && b > 0) then
        q,r
    else if (a >= 0 && b < 0) then
```

1. l'existence de la décomposition en base -2 est claire et la suite r est déterminée de manière unique, par unicité de la division euclidienne dans $\mathbb{Z}$.

```
        -q,r
    else if (a <= 0 && b > 0) then
        -q-1,b-r
    else
        q+1,-b-r;;

let rec Polynome_negabinaire n = match (div_eucl n (-2)) with
        | 0, r -> [r]
        | q, r -> r::(Polynome_negabinaire q);;

Polynome_negabinaire(19);;
```

SORTIE

```
#div_eucl : int -> int -> int * int = <fun>
#Polynome_negabinaire : int -> int list = <fun>
#- : int list = [1; 1; 1; 0; 1]
```