Corrigé

option informatique

Détection de cycles

Question 1. La première version de la fonction **itere** utilise le fait que $x_n = f^n(x_0) = f \circ f^{n-1}(x_0)$, la seconde que $x_n = f^n(x_0) = f^{n-1} \circ f(x_0)$. Nous apprendrons plus tard que cette seconde version possède un avantage sur la première.

```
let rec itere1 f x0 = function
  | 0 -> x0
  | n -> f(itere1 f x0 (n-1)) ;;
```

```
let rec itere2 f x0 = function
  | 0 -> x0
  | n -> itere2 f (f x0) (n-1) ;;
```

Question 2. On montre sans peine que pour tout $n \in \mathbb{N}$, $y_n = x_{2n}$. Mais :

$$x_{2n} = x_n \iff (n \geqslant \mu \text{ et } 2n \equiv n \bmod \lambda) \iff (n \geqslant \mu \text{ et } \lambda \text{ divise } n)$$

donc l'algorithme de Floyd se termine et retourne le plus petit multiple de λ qui soit supérieur ou égal à μ.

```
let floyd1 f x0 =
  let rec aux = fun
    | x y when x = y -> x
    | x y            -> aux (f x) (f (f y))
  in aux (f x0) (f (f x0)) ;;
```

```
let floyd2 f x0 =
  let rec aux i = fun
    | x y when x = y -> i
    | x y            -> aux (i+1) (f x) (f (f y))
  in aux 1 (f x0) (f (f x0)) ;;
```

La suite des itérés de x_i est une suite périodique : sa pré-période est nulle et sa période égale à λ ; la fonction **floyd2** appliquée à x_i retourne donc la valeur de λ. D'où la fonction :

```
let periode f x0 =
  let xi = floyd1 f x0 in floyd2 f xi ;;
```

Puisque i est un multiple de λ on a $x_{i+\mu} = x_\mu$. Il suffit donc pour trouver μ de comparer les suites des itérés de x_0 et de x_i jusqu'à trouver une valeur commune.

```
let pre_periode f x0 =
  let xi = floyd1 f x0 in
  let rec aux mu = fun
    | x y when x = y -> mu
    | x y            -> aux (mu + 1) (f x) (f y)
  in aux 0 x0 xi ;;
```

Question 3. On définit la fonction :

```
let brent f x0 =
  let rec aux (i, j) = fun
    | x y when x = y -> (i, j)
    | x y when j <= 2 * i -> aux (i, j+1) x (f y)
    | x y                 -> aux (j, j+1) y (f y)
  in aux (0, 1) x0 (f x0) ;;
```

On prouve sans peine que les valeurs prises par i sont les entiers de la forme $i = 2^n - 1$ et que pour un tel entier j prend toutes les valeurs de l'intervalle $[\![2^n, 2^{n+1} - 1]\!]$.
Mais $x_i = x_j \iff (i \geqslant \mu$ et λ divise $j - i)$, avec $1 \leqslant j - i \leqslant 2^n$. Notons donc n_0 le plus petit entier vérifiant $2^n - 1 \geqslant \mu$ et $2^n \geqslant \lambda$. L'algorithme de Brent se termine lorsque $i = 2^{n_0} - 1$ et $j = \lambda + i$.
Il y a deux avantages à appliquer l'algorithme de Brent plutôt que celui de Floyd : il n'est nécessaire que de faire un seul parcours pour obtenir la valeur de λ puisqu'une fois obtenues les valeurs de i et j on a $\lambda = j - i$, et en outre, à chaque étape un seul calcul de f est effectué, contre trois pour l'algorithme de Floyd.