TP OCaml basique

Dans ce TP d'introduction, toutes les variables sont persistantes, aucune liste n'intervinent et la récursion n'est pas nécessairement terminale.

Exercice 1. écrire une fonction `sigma : (int -> int) -> int -> int` qui prend en paramètre une fonction f et retourne la somme

$$\Sigma_{i=0}^{n} f(i).$$

```
# sigma (fun x -> x*x) 3;;
- : int = 14
```

Exercice 2.

1. Écrire une fonction `incr : int -> int -> int` qui renvoie $x+1$ si $a > 0$, x-1 si $a < 0$ et x sinon.

```
# incr 3 6, incr 0 6, incr (-8) 6;;
- : int * int * int = (7, 6, 5)
```

2. Écrire une fonction `somme : int -> int -> int` qui retourne la somme de deux nombres. Aucun symbole `+` ou `-` ne doit être utilisé.
   La fonction doit avoir un coût linéaire.

Exercice 3. Écrire une fonction `prod: int -> int -> int` qui renvoie le produit de deux entiers positifs. Les seules opérateurs arithmétiques autorisés sont `+` et `/`.
La fonction doit avoir un coût logarithmique.

Exercice 4. On rappelle que pour deux nombres entiers a, b, on a $\text{pgcd}(a,b) = \text{pgcd}(b,r)$ où r est le reste de la division euclidienne de a par b. De plus, par convention, $\text{pgcd}(a,0) = 0$.
Ces deux observations sont au cœur de l'*algorithme d'Euclide* pour la recherche du pgcd.

1. Écrire une fonction OCaml `pgcd : int -> int -> int` qui calcule par l'algorithme d'Euclide le pgcd de deux nombres.

```
# pgcd 21 15;;
- : int = 3
# pgcd (3*3*2*5*7*7*7*11) (3*3*3*3*5*7);;
- : int = 315
```

2. On admet que si a, b sont deux entiers positifs tels que $a \geq b > 0$, alors $a\%b \leq \lfloor \frac{a}{2} \rfloor$ en notant % l'opérateur de calcul du reste.
   (a) On suppose a, b écrits sur n bits et $a > b$ (donc b est en fait écrit sur moins de n bits). Majorer le nombre total d'appels à `pgcd` pour l'appel initial `pgcd a b` en fonction de n.

(b) Majorer le nombre total d'appels à `pgcd` pour l'appel initial `pgcd a b` en fonction de a.

(c) On admet que la recherche du reste pour deux nombre écrits sur au plus n bits est majorée par un $O(n^2)$ opérations élémentaires ; que la comparaison à 0 est en $O(n)$ (il y a n bits à comparer) et que les écritures (comme celle à l'adresse de retour) sont majorées par un $O(n)$ (il y a n bits à écrire).

Donner, en fonction de a, une majoration pas trop grossière de la complexité au pire de l'appel `pgcd a b` (on ne prend en compte que les calculs de restes, les écritures et les compraisons à 0).