Exercice 1. Terminaison

Montrer que la fonction suivante termine :

```
let rec f a b =
    if a = 0 || b = 0 then 1
    else if a mod 2 = 0 then f (a/3) (2*b)
    else f (2*a) (b/3)
```

Exercice 2. Invariant de boucle simple

En utilisant un invariant de boucle, prouver que la fonction suivante renvoie bien la somme des éléments d'un tableau :

```
let somme t =
    let s = ref 0 in
    for i = 0 to Array.length t - 1 do
        s := !s + t.(i)
    done;
    !s
```

On rappelle qu'un invariant de boucle est une propriété qui reste vraie à chaque itération de la boucle et qui permet de montrer que la fonction renvoie le bon résultat (ici, la somme des éléments de `t`).

On prouve cet invariant de boucle par récurrence sur le nombre d'itérations dans la boucle.

Exercice 3. Encore l'algorithme d'Euclide

Prouver par récurrence que l'algorithme d'Euclide renvoie bien le pgcd de deux nombres :

```
let rec pgcd a b =
    if b = 0 then a
    else pgcd b (a mod b)
```

Exercice 4. Tranche maximum (algorithme de Kadane)

On considère un tableau `t` d'entiers.

Une **somme consécutive** (ou tranche) dans `t` est de la forme $\sum_{k=i}^{j}$ `t.(k)` (où i et j sont des indices de `t`).

On note s la valeur maximum d'une somme consécutive.

1. Écrire une fonction `tranche_max` prenant `t` en argument et renvoyant `s`, en complexité quadratique en la taille de `t`.

Si j est un indice de `t`, on note s_j la plus grande somme consécutive finissant en j. Dit autrement :

$$s_j = \max_{0 \leq i \leq j} \sum_{k=i}^{j} t.(k)$$

1. Calculer tous les s_j, si `t = [|1; 4; -3; 5; -7; 0|]`
2. Si $j > 0$, montrer que :
$$s_j = \max(s_{j-1} + t.(j),\ t.(j))$$
3. Comment peut-on exprimer s en fonction de s_j ?
4. En déduire une fonction `tranche_max` prenant `t` en argument et renvoyant `s`, en complexité linéaire en la taille de `t`.
5. Modifier votre fonction précédente pour obtenir les indices de début et fin de s.