Correction TD 1 algorithmes récursifs

1. `f x y` calcule le produit des entiers x et y. La correction se prouve par récurrence sur la valeur absolue de x.

2. Pour n entier naturel, `g n` calcule le logarithme binaire de n, c'est-à-dire l'entier k tel que $2^k \leqslant n < 2^{k+1}$. La correction et la terminaison se prouvent par récurrence sur n.

3.
```
let rec est_palindrome s = match String.length s with
  0 -> true
 | 1 -> true
 | n -> (s.[0] = s.[n-1]) && (est_palindrome (String.sub s 1 (n-2)))
;;
```

4.
```
let rec binaire n = match n with
  1 -> print_int 1
 | n -> binaire (n/2); print_int (n mod 2) ;;
```

5.
```
let rec numero = function
   (0,0) -> 0
 | (a,0) -> 1 + numero (0,a-1)
 | (a,b) -> 1 + numero (a+1,b-1)
;;
let rec couple  = function
   0 -> (0,0)
 | n -> let (a,b) = couple (n-1) in
        if a = 0 then (b+1,0) else (a-1,b+1)  ;;
```

   On démontre par induction structurelle sur l'ordre lexicographique de $\mathbb{N}^2$ que $f(p,q) = \dfrac{(p+q)(p+q+1)}{2} + q$.

6. On écrit une fonction auxiliaire aux telle que `aux i` renvoie un couple `(ming,cmin)` où `ming` est la coupe minimale de `t[i..n-1]` (n longueur du tableau) commençant à l'indice i et `cmin` est la coupe minimale de `t[i..n-1]`. On calcule récursivement `aux i` à partir de `aux (i+1)`, le cas de base correspondant à la dernière case du tableau. La coupe maximale s'obtient par `aux 0`.

```
let coupe t =
  let rec aux i =
   if i = Array.length t -1  then t.(i),t.(i)
   else begin
      let (ming,cmin) = aux (i+1) in
      let newming = min t.(i) (t.(i)+ming) in
      (newming, min newming cmin)
      end
  in snd (aux 0) ;;
```

7. (a) On écrit une fonction récursive auxiliaire `ana_aux` telle que `ana_aux a b` affiche la concaténation de `b` avec les anagrammes de `a`.

```
let anagrammes mot =
 let rec ana_aux a b =
  let n = String.length a in
  if n = 1 then (print_string (b^a); print_string " ")
  else begin
   for k = 0 to n-1 do
```

```
    ana_aux ((String.sub a 0 k)^(String.sub a (k+1) (n-1-k)))
             (b^(String.sub a k 1)) done
    end
  in ana_aux mot "" ;;
```

(b) On suppose que le mot m est de longueur n, formée des lettres distinctes $a_1, \dots, a_p$, respectivement utilisées $\alpha_1, \dots, \alpha_p$ fois. Le nombres d'anagrammes distinctes de m est égal à $\frac{n!}{\alpha_1!\,\alpha_2!\,\cdots\,\alpha_p!}$.

8.
- ```
  let rec t n = match n with
    0 -> 0
   | n when n mod 2 = 0 -> t(n/2)
   | n -> 1 - t(n/2) ;;
  ```
- t_n est la somme modulo 2 des chiffres du développement binaire de n (preuve par récurrence).
- ```
  let rec compte = function
     0 -> 0
   | n -> if t n = 0 then compte (n-1) else 1 + compte (n-1) ;;
  ```
- Le mot infini formé de la suite de Thue-Morse ne contient aucun cube.
  Le mot infini formé de la suite t_{n^2} est normale, c'est-à-dire que tous les facteurs apparaissent, et deux facteurs de même longueur apparaissent avec la même fréquence asymptotique.

  ```
  let liste_tn2 n =
    let rec liste = function
      0 -> [0]
     | n -> (t (n*n)) :: liste(n-1)
    in List.rev (liste n)
  ;;
  let liste_tn m n =
    let rec liste = function
       n when n = m -> [t m]
     | n -> (t n) :: liste(n-1)
    in List.rev (liste n)
  ;;
  ```