cas des arbres binaires homogènes

```
type 'a abho =
    n_abho of 'a abho * 'a * 'a abho     (* definition des noeuds *)
|   f_abho of 'a;;                              (* définition des feuilles *)

let rec hauteur_abho = function
               f_abho(x) -> 0
 |   n_abho(fg,x,fd) -> 1 + max (hauteur_abho fg) (hauteur_abho fd);;

let rec nbnoeuds_abho = function
               f_abho(x) -> 0
 |   n_abho(fg,x,fd) -> 1 + (nbnoeuds_abho fg) + (nbnoeuds_abho fd);;

let rec nbfeuilles_abho = function
             f_abho(x)   -> 1
 |   n_abho(fg,x,fd) -> (nbfeuilles_abho fg) + (nbfeuilles_abho fd);;

let rec estfeuille_abho x = function
     f_abho(y) when x=y -> true
 |   f_abho(y)                -> false
 |   n_abho(fg,y,fd)       -> (estfeuille_abho x fg) or (estfeuille_abho x fd);;

let rec estnoeud_abho x = function
     f_abho(y)           -> false
 |   n_abho(fg,y,fd) when x=y -> true
 |   n_abho(fg,y,fd)  -> (estnoeud_abho x fg) or (estnoeud_abho x fd);;

let rec remplace_abho x  y = function
     f_abho(z) when z=x  -> f_abho(y)
 |   f_abho(z)                -> f_abho(z)
 |   n_abho(fg,z,fd) when z=x -> n_abho(remplace_abho x y fg,y,remplace_abho x y fd)
 |   n_abho(fg,z,fd)        -> n_abho(remplace_abho x y fg,z,remplace_abho x y fd) ;;
```

cas des arbres binaires hétérogènes.

```
type ('a,'b) abhe =
    n_abhe of ('a,'b) abhe * 'a * ('a,'b) abhe     (* definition des noeuds *)
|   f_abhe of 'b;;                              (* définition des feuilles *)

let rec hauteur_abhe = function
               f_abhe(x) -> 0
 |   n_abhe(fg,x,fd) -> 1 + max (hauteur_abhe fg) (hauteur_abhe fd);;

let rec nbnoeuds_abhe = function
               f_abhe(x) -> 0
 |   n_abhe(fg,x,fd) -> 1 + (nbnoeuds_abhe fg) + (nbnoeuds_abhe fd);;

let rec nbfeuilles_abhe = function
             f_abhe(x)   -> 1
 |   n_abhe(fg,x,fd) -> (nbfeuilles_abhe fg) + (nbfeuilles_abhe fd);;

```

```
let rec estfeuille_abhe x = function
     f_abhe(y) when x=y -> true
  |  f_abhe(y)                -> false
  |  n_abhe(fg,y,fd)       -> (estfeuille_abhe x fg) or (estfeuille_abhe x fd);;

let rec estnoeud_abhe x = function
     f_abhe(y)           -> false
  |  n_abhe(fg,y,fd) when x=y -> true
  |  n_abhe(fg,y,fd)  -> (estnoeud_abhe x fg) or (estnoeud_abhe x fd);;

let rec remplacef_abhe x y = function
     f_abhe(z) when z=x  -> f_abhe(y)
  |  f_abhe(z)                -> f_abhe(z)
  |  n_abhe(fg,z,fd)        -> n_abhe(remplacef_abhe x y fg,z,remplacef_abhe x y fd) ;;

let rec remplacen_abhe x y = function
     f_abhe(z)                -> f_abhe(z)
  |  n_abhe(fg,z,fd) when z=x -> n_abhe(remplacen_abhe x y fg,y,remplacen_abhe x y fd)
  |  n_abhe(fg,z,fd)          -> n_abhe(remplacen_abhe x y fg,z,remplacen_abhe x y fd) ;;
```

cas des arbres ternaires homogènes

```
type 'a atho =
    n_atho of 'a * 'a atho * 'a atho* 'a atho     (* definition des noeuds *)
|   f_atho of 'a;;                              (* définition des feuilles *)

let rec hauteur_atho = function
               f_atho(x) -> 0
  |  n_atho(x,f1,f2,f3) -> 1 + max (hauteur_atho f1) (max (hauteur_atho f2) (hauteur_atho f3));;

let rec nbnoeuds_atho = function
               f_atho(x) -> 0
  |  n_atho(x,f1,f2,f3) -> 1 + (nbnoeuds_atho f1) + (nbnoeuds_atho f2) + (nbnoeuds_atho f3);;

let rec nbfeuilles_atho = function
             f_atho(x)   -> 1
  |  n_atho(x,f1,f2,f3) -> (nbfeuilles_atho f1) + (nbfeuilles_atho f2) + (nbfeuilles_atho f3);;

let rec estfeuille_atho x = function
     f_atho(y) when x=y -> true
  |  f_atho(y)                -> false
  |  n_atho(y,f1,f2,f3)       -> (estfeuille_atho x f1) or (estfeuille_atho x f2) or (estfeuille_

let rec estnoeud_atho x = function
     f_atho(y)           -> false
  |  n_atho(y,f1,f2,f3) when x=y -> true
  |  n_atho(y,f1,f2,f3)  -> (estnoeud_atho x f1) or (estnoeud_atho x f2) or (estnoeud_atho x f2);

let rec remplace_atho x  y = function
     f_atho(z) when z=x  -> f_atho(y)
  |  f_atho(z)           -> f_atho(z)
  |  n_atho(z,f1,f2,f3) when z=x -> n_atho(y,remplace_atho x y f1,remplace_atho x y f2,remplace_a
  |  n_atho(z,f1,f2,f3)          -> n_atho(z,remplace_atho x y f1,remplace_atho x y f2,remplace_a
```

cas des arbres ternaires hétérogènes

```
type ('a,'b) athe =
    n_athe of 'a * ('a,'b) athe * ('a,'b) athe * ('a,'b) athe     (* définition des noeuds *)
|   f_athe of 'b;;                                (* définition des feuilles *)

let rec hauteur_athe = function
              f_athe(x) -> 0
 |   n_athe(x,f1,f2,f3) -> 1 + max (hauteur_athe f1) (max (hauteur_athe f2) (hauteur_athe f3));;

let rec nbnoeuds_athe = function
              f_athe(x) -> 0
 |   n_athe(x,f1,f2,f3) -> 1 + (nbnoeuds_athe f1) + (nbnoeuds_athe f2) + (nbnoeuds_athe f3);;

let rec nbfeuilles_athe = function
            f_athe(x)   -> 1
 |   n_athe(x,f1,f2,f3) -> (nbfeuilles_athe f1) + (nbfeuilles_athe f2) + (nbfeuilles_athe f3);;

let rec estfeuille_athe x = function
     f_athe(y) when x=y -> true
 |   f_athe(y)                -> false
 |   n_athe(y,f1,f2,f3)       -> (estfeuille_athe x f1) or (estfeuille_athe x f2) or (estfeuille_

let rec estnoeud_athe x = function
     f_athe(y)           -> false
 |   n_athe(y,f1,f2,f3) when x=y -> true
 |   n_athe(y,f1,f2,f3)  -> (estnoeud_athe x f1) or (estnoeud_athe x f2) or (estnoeud_athe x f3);

let rec remplacef_athe x y = function
     f_athe(z) when z=x  -> f_athe(y)
 |   f_athe(z)                -> f_athe(z)
 |   n_athe(z,f1,f2,f3)       -> n_athe(z,remplacef_athe x y f1,remplacef_athe x y f2,remplacef_

let rec remplacen_athe x y = function
     f_athe(z)               -> f_athe(z)
 |   n_athe(z,f1,f2,f3) when z=x -> n_athe(y,remplacen_athe x y f1,remplacen_athe x y f2,remplace
 |   n_athe(z,f1,f2,f3)          -> n_athe(z,remplacen_athe x y f1,remplacen_athe x y f2,remplace
```

cas des arbres homogènes où les noeuds sont représentés par des listes d'arbres

```
type 'a alho =
    n_alho of 'a * 'a alho list     (* definition des noeuds *)
|   f_alho of 'a;;                              (* définition des feuilles *)

let rec hauteur_alho = function
              f_alho(x) -> 0
 |   n_alho(x,lf) -> 1 + (it_list max 0 (map hauteur_alho lf));;

let rec nbnoeuds_alho = function
              f_alho(x) -> 0
 |   n_alho(x,lf) -> 1 + (it_list add_int 0 (map nbnoeuds_alho lf));;

let rec nbfeuilles_alho = function
            f_alho(x)   -> 1
 |   n_alho(x,lf) -> it_list add_int 0 (map nbfeuilles_alho lf);;

let rec ou x y = x or y;;
```

```
18
19  let rec estfeuille_alho x = function
20       f_alho(y) when x=y -> true
21    |  f_alho(y)          -> false
22    |  n_alho(y,lf)       -> it_list ou false (map (estfeuille_alho x) lf);;
23
24  let rec estnoeud_alho x = function
25       f_alho(y)           -> false
26    |  n_alho(y,lf) when x=y -> true
27    |  n_alho(y,lf)  -> it_list ou false (map (estnoeud_alho x) lf);;
28
29  let rec remplace_alho x  y = function
30       f_alho(z) when z=x  -> f_alho(y)
31    |  f_alho(z)           -> f_alho(z)
32    |  n_alho(z,lf) when z=x -> n_alho(y,map (remplace_alho x y) lf)
33    |  n_alho(z,lf)          -> n_alho(z,map (remplace_alho x y) lf);;
```

cas des arbres hétérogènes où les noeuds sont représentés par des listes d'arbres

```
1   type ('a,'b) alhe =
2       n_alhe of 'a * ('a,'b) alhe list     (* définition des noeuds *)
3   |   f_alhe of 'b;;                              (* définition des feuilles *)
4
5   let rec hauteur_alhe = function
6                 f_alhe(x) -> 0
7    |   n_alhe(x,lf) -> 1 + (it_list max 0 (map hauteur_alhe lf));;
8
9   let rec nbnoeuds_alhe = function
10                f_alhe(x) -> 0
11   |   n_alhe(x,lf) -> 1 + (it_list add_int 0 (map nbnoeuds_alhe lf));;
12
13  let rec nbfeuilles_alhe = function
14              f_alhe(x)   -> 1
15   |   n_alhe(x,lf) -> it_list add_int 0 (map nbfeuilles_alhe lf);;
16
17  let rec ou x y = x or y;;
18
19  let rec estfeuille_alhe x = function
20       f_alhe(y) when x=y -> true
21    |  f_alhe(y)          -> false
22    |  n_alhe(y,lf)       -> it_list ou false (map (estfeuille_alhe x) lf);;
23
24  let rec estnoeud_alhe x = function
25       f_alhe(y)           -> false
26    |  n_alhe(y,lf) when x=y -> true
27    |  n_alhe(y,lf)  -> it_list ou false (map (estnoeud_alhe x) lf);;
28
29  let rec remplacen_alhe x  y = function
30       f_alhe(z)           -> f_alhe(z)
31    |  n_alhe(z,lf) when z=x -> n_alhe(y,map (remplacen_alhe x y) lf)
32    |  n_alhe(z,lf)          -> n_alhe(z,map (remplacen_alhe x y) lf);;
33
34  let rec remplacef_alhe x y = function
35       f_alhe(z) when z=x  -> f_alhe(y)
36    |  f_alhe(z)           -> f_alhe(z)
37    |  n_alhe(z,lf)          -> n_alhe(z,map (remplacef_alhe x y) lf);;
```

cas des arbres homogènes où les noeuds sont représentés par des vecteurs d'arbres

```
type 'a avho =
    n_avho of 'a * 'a avho vect     (* definition des noeuds *)
|   f_avho of 'a;;                            (* définition des feuilles *)

let rec hauteur_avho = function
        f_avho(x) -> 0
 |   n_avho(x,lf) -> 1 + (it_list max 0 (map_vect_list hauteur_avho lf));;

let rec nbnoeuds_avho = function
        f_avho(x) -> 0
 |   n_avho(x,lf) -> 1 + (it_list add_int 0 (map_vect_list nbnoeuds_avho lf));;

let rec nbfeuilles_avho = function
      f_avho(x)   -> 1
 |   n_avho(x,lf) -> it_list add_int 0 (map_vect_list nbfeuilles_avho lf);;

let rec ou x y = x or y;;

let rec estfeuille_avho x = function
     f_avho(y) when x=y -> true
 |   f_avho(y)          -> false
 |   n_avho(y,lf)       -> it_list ou false (map_vect_list (estfeuille_avho x) lf);;

let rec estnoeud_avho x = function
     f_avho(y)           -> false
 |   n_avho(y,lf) when x=y -> true
 |   n_avho(y,lf)  -> it_list ou false (map_vect_list (estnoeud_avho x) lf);;

let rec remplace_avho x  y = function
     f_avho(z) when z=x    -> f_avho(y)
 |   f_avho(z)             -> f_avho(z)
 |   n_avho(z,lf) when z=x -> n_avho(y,map_vect (remplace_avho x y) lf)
 |   n_avho(z,lf)          -> n_avho(z,map_vect (remplace_avho x y) lf);;
```

cas des arbres hétérogènes où les noeuds sont représentés par des vecteurs d'arbres

```
type ('a,'b) avhe =
    n_avhe of 'a * ('a,'b) avhe vect     (* définition des noeuds *)
|   f_avhe of 'b;;                       (* définition des feuilles *)

let rec hauteur_avhe = function
              f_avhe(x) -> 0
 |   n_avhe(x,lf) -> 1 + (it_list max 0 (map_vect_list hauteur_avhe lf));;

let rec nbnoeuds_avhe = function
        f_avhe(x) -> 0
 |   n_avhe(x,lf) -> 1 + (it_list add_int 0 (map_vect_list nbnoeuds_avhe lf));;

let rec nbfeuilles_avhe = function
      f_avhe(x)   -> 1
 |   n_avhe(x,lf) -> it_list add_int 0 (map_vect_list nbfeuilles_avhe lf);;

let rec ou x y = x or y;;


```

```
let rec estfeuille_avhe x = function
     f_avhe(y) when x=y -> true
  |  f_avhe(y)          -> false
  |  n_avhe(y,lf)       -> it_list ou false (map_vect_list (estfeuille_avhe x) lf);;

let rec estnoeud_avhe x = function
       f_avhe(y)           -> false
  |  n_avhe(y,lf) when x=y -> true
  |  n_avhe(y,lf)  -> it_list ou false (map_vect_list (estnoeud_avhe x) lf);;

let rec remplacen_avhe x  y = function
       f_avhe(z)           -> f_avhe(z)
  |  n_avhe(z,lf) when z=x -> n_avhe(y,map_vect (remplacen_avhe x y) lf)
  |  n_avhe(z,lf)          -> n_avhe(z,map_vect (remplacen_avhe x y) lf);;

let rec remplacef_avhe x y = function
     f_avhe(z) when z=x  -> f_avhe(y)
  |  f_avhe(z)           -> f_avhe(z)
  |  n_avhe(z,lf)        -> n_avhe(z,map_vect (remplacef_avhe x y) lf);;
```

Exercice n°3

```
let rec somme_abho = function
   f_abho(x)      -> x
| n_abho(fg,x,fd) -> (somme_abho fg) + x + (somme_abho fd);;

let rec max_abho = function
    f_abho(x)     -> x
| n_abho(fg,x,fd) -> max (max_abho fg) (max x  (max_abho fd));;

let rec somme_atho = function
     f_atho(x)       -> x
| n_atho(x,f1,f2,f3) -> x + (somme_atho f1) + (somme_atho f2) + (somme_atho f3);;

let rec max_atho = function
    f_atho(x)        -> x
| n_atho(x,f1,f2,f3) -> max (max x (max_atho f1)) (max (max_atho f2) (max_atho f3));;

let rec somme_alho = function
   f_alho(x)   -> x
| n_alho(x,lf) -> it_list add_int x (map somme_alho lf);;

let rec max_alho = function
   f_alho(x)   -> x
| n_alho(x,lf) ->  it_list max  x (map max_alho lf);;

let rec somme_avho = function
   f_avho(x)   -> x
| n_avho(x,lf) -> it_list add_int x (map_vect_list somme_avho lf);;

let rec max_avho = function
   f_avho(x)   -> x
| n_avho(x,lf) -> it_list max  x (map_vect_list max_avho lf);;
```