Corrigé feuille d'exercice n°10

(* Déclaration de type et ouverture de la fenêtre graphique *)

```
#open "graphics";;

open_graph "";;

type circuit = AND of circuit * circuit
                    | OR    of circuit * circuit
                    | NOT of circuit
                    | Entrée of string;;
```

(* Préliminaires *)
(* fonction pour dessiner une porte logique *)

```
let dessine_carre x y l =
     let d  = l/2 in
          moveto (x-d) (y-d);
          lineto (x-d) (y+d);
          lineto (x+d) (y+d);
          lineto (x+d) (y-d);
          lineto (x-d) (y-d);;

let dessine x y porte =
   dessine_carre x y 20;
   match porte with
      AND(a,b) -> moveto (x-8) (y-5); draw_string "AND"
   |  OR (a,b)  -> moveto (x-5) (y-4); draw_string "OR"
   | NOT(a)    -> moveto (x-8) (y-5); draw_string "NOT"
   |   _    -> failwith "erreur";;

dessine 150 50 (AND(Entrée("toto"),Entrée("toto")));;

let ligne_brisée (x1,y1) (x2,y2) xi =
      moveto x1 y1;
      lineto xi y1;
      lineto xi y2;
      lineto x2 y2;;

ligne_brisée (100,100) (150,130) 120;;
```

(* fonction classique de calcul de hauteur d'arbre *)

```
let rec hauteur = function
   Entrée(s) -> 0
| NOT(c) -> 1 + hauteur c
| OR(c1,c2) -> 1 + max (hauteur c1) (hauteur c2)
| AND(c1,c2) -> 1 + max (hauteur c1) (hauteur c2);;
```

(* fonction classique de calcul de largeur d'arbre *)

```
let rec largeur = function
   Entrée(s) -> 1
| NOT(c) -> largeur c
| OR(c1,c2) -> (largeur c1) + (largeur c2)
| AND(c1,c2) ->  (largeur c1) + (largeur c2);;
```

(* fonction de parcours d'arbres pour collecter les entrées du circuit sous forme de liste *)

```
let rec appart x = function
  [] -> false
| y::ys when x=y -> true
| y::ys -> appart x ys;;

let parcours circ =
   let rec parcourir l = function
       Entrée(s) when appart s l -> l
     | Entrée(s) -> s::l
     | NOT(c) -> parcourir l c
     | OR(c1,c2) -> union (parcourir l c1) (parcourir l c2)
     | AND(c1,c2) ->  union (parcourir l c1) (parcourir l c2) in
   parcourir [] circ;;
```

(* Dessin d'un circuit *)
(* fonction pour dessiner un circuit logique représenté sous forme d'arbre *)

```
let dessine_circuit c =
    let xmax = size_x () and ymax = size_y ()
    and liste_entrées = parcours c in
        let nb_entrées = list_length liste_entrées
    and p = hauteur c
    in let larg = xmax / (p+2) in
       let xi_entrée s =
           50+10*(index s liste_entrées) in
    let rec dess circ ym yM prof =
                match circ with
      AND(c1,c2) -> let x0 = prof * larg and y0 = (ym + yM) / 2 in
                    dessine   x0 y0 (AND(Entrée(""),Entrée("")));
                        let l1 = largeur c1 and l2 = largeur c2 in
                        let y1 = ym + (yM -ym)*l1/(l1+l2) in
                             let (x1,y1) = dess c1 ym y1 (prof-1) and
                                 (x2,y2) = dess c2 y1 yM (prof-1) in
                                 let xi1 = (match c1 with Entrée(s) -> xi_entrée(s)
                                                        | _ -> x0 -larg /2) and
                                     xi2 = (match c2 with Entrée(s) -> xi_entrée(s)
                                                        | _ -> x0 -larg /2) in
                                   ligne_brisée (x1,y1) (x0-10,y0-3) (xi1);
                                   ligne_brisée (x2,y2) (x0-10,y0+3) (xi2);
                                   (x0+10, y0)
  |   OR(c1,c2) -> let x0 = prof * larg and y0 = (ym + yM) / 2 in
                   dessine   x0 y0 (OR(Entrée(""),Entrée("")));
                   let l1 = largeur c1 and l2 = largeur c2 in
                        let y1 = ym + (yM -ym)*l1/(l1+l2) in
                             let (x1,y1) = dess c1 ym y1 (prof-1) and
                                 (x2,y2) = dess c2 y1 yM (prof-1) in
                                 let xi1 = (match c1 with Entrée(s) -> xi_entrée(s)
                                                        | _ -> x0 -larg /2) and
                                     xi2 = (match c2 with Entrée(s) -> xi_entrée(s)
                                                        | _ -> x0 -larg /2) in
                                   ligne_brisée (x1,y1) (x0-10,y0-3) (xi1);
                                   ligne_brisée (x2,y2) (x0-10,y0+3) (xi2);
                                   (x0+10, y0)
  |  NOT(c1) -> let x0 = prof * larg and y0 = (ym + yM) / 2 in
                   dessine   x0 y0 (NOT(Entrée("")));
                             let (x1,y1) = dess c1 ym yM (prof-1)  in
                                   let xi = (match c1 with Entrée(s) -> xi_entrée(s)
                                                         | _ -> x0 -larg /2) in
                                   ligne_brisée (x1,y1) (x0-10,y0) xi;
                                   (x0+10, y0)
  |  Entrée(s) -> let nb = index s liste_entrées in
                       moveto 10 (ymax*nb/(nb_entrées+1)+50);
                       draw_string s;
                       current_point () in

    dess c 0 ymax (hauteur c);;

clear_graph ();;

dessine_circuit (AND(OR(Entrée "a", NOT(Entrée "b")), Entrée "b"));;

clear_graph ();;

let A = Entrée "a";;
let B = Entrée "b";;
let CIN = Entrée "cin";;
let add = OR(AND(OR(AND(NOT A, B),AND(A,NOT B)),NOT CIN),AND(OR(AND(A,B),AND(NOT A,NOT B)), CIN));;

dessine_circuit add;;
```