# Programmation : Correction TP 11 noté (S03, 23 mars 2004)

## [C] Quadrature

1. Fonction qui calcule, pour un nombre $n$, le nombre d'itérations nécessaire pour retrouver un nombre déjà vu. Le tableau comporte 100 cases et est indexé par les nombres générés à partir des $2^e$ et $3^e$ chiffres. Ainsi, dès que l'on obtient un nombre pour la première fois, on passe la case du tableau qui lui correspond à `TRUE`.

```
#include <stdlib.h>
#include <stdio.h>

#define FALSE 0
#define TRUE 1

#define CentainesEtDizaines(u) (((u) / 10) % 100)

int calc (int u) {
  int p[100];
  int i;
  int nb_iter = 0;
  int new_nb = CentainesEtDizaines(u);
  for (i=0; i < 100; i++) {  /* initialisation du tableau */
    p[i] = TRUE;
  }
  while (p[new_nb]) {
    p[new_nb] = FALSE; /* le nombre new_b est marqué déjà obtenu */
    new_nb = CentainesEtDizaines(new_nb * new_nb);
    nb_iter++;
  }
  return (nb_iter);
}
```

2. Pour cette question, il suffit de réutiliser la fonction ci-dessus en l'appelant avec $(u_0)^2$ mais au lieu de renvoyer le nombre d'itérations, on retourne la valeur du nombre déjà obtenu. En effet, pour une telle suite, 2 cas sont possibles : soit le nombre déjà obtenu est nul donc la suite converge vers 0, soit le nombre déjà obtenu est non nul et la suite boucle indéfiniment.

```
int calc (int u) {
  ... /* même chose que précédemment */
  return (new_nb);
}

int nb_suite() {
  int i;
  int nb = 0;
  for (i=0;i<100;i++)
    if (calc(i*i) == 0)
      nb++;
  return nb;
}
```

## [OCaml] Représentation d'un système de fichiers

1. Définition d'un type permettant de représenter un système de fichier :

```
#type size = int;;

#type name = string;;

#type file_system =
#    Dir of name * file_system array
#  | File of name * size;;
```

2. Représentation d'une arborescence :

```
#let arbre =
#  Dir ("racine",
#   [|File ("cuisine.txt",1000);
#     Dir ("C",[|File ("tp1.c",200);File ("tp2.c",180)|]);
#     Dir ("OCaml",[|Dir ("tp1",[|File ("tp1.ml",100)|]);File ("tp2.ml",50)|]);
#   |]);;
```

3. Fonction qui permet de lister le contenu d'un système de fichier :

```
#let ls = fun arborescence ->
#  let print_space = fun n -> (* pour une jolie mise en page *)
#    for i=1 to n do
#      Printf.printf "  "
#    done in
#  let rec aff = fun arbre prof ->
#    match arbre with
#      File (nom, _taille) -> print_space prof; Printf.printf "%s\n" nom
#    | Dir (nom, ss_arbre) ->
#        print_space prof; Printf.printf "%s/\n" nom;
#        Array.iter
#          (fun ss -> aff ss (prof+1))
#          ss_arbre in
#  aff arborescence 0;;
val ls : file_system -> unit = <fun>
```

4. Fonction qui permet d'afficher la taille de chaque répertoire d'un système de fichier.

```
#let du = fun arborescence ->
#  let rec calc = fun arbre pwd ->
#    match arbre with
#      File (_nom, taille) -> taille (* renvoie la taille du fichier *)
#    | Dir (nom, ss_arbre) ->
#        let sum = ref 0
#        and pwd' = pwd ^ "/" ^ nom in
#        Array.iter (* somme les tailles du contenu du repertoire *)
#          (fun ss -> sum := !sum + (calc ss pwd'))
#          ss_arbre;
#        Printf.printf "%d\t%s\n" !sum pwd';
#        !sum in
#  ignore (calc arborescence ".");;
val du : file_system -> unit = <fun>
```