Le Bezvoët Evelyne

# Projet programmation

my_list.ml

```
type 'a my_list =
        | Nil
        | Cons of 'a * 'a my_list

let string_of_list str_fun l =
        let rec string_content = function
                  | Nil -> ""
                  | Cons (x,Nil) -> (str_fun x)
                  | Cons (x,l) -> (str_fun x) ^ ", " ^ (string_content l)
                          in "[" ^ (string_content l) ^ "]";;

let hd = function
        | Nil -> None
        | Cons (x,l) -> Some x;;

let tl = function
        | Nil -> None
        | Cons (x,l) -> Some l;;

let rec length = function
        | Nil -> 0
        | Cons (x,l) -> 1 + (length l);;

let rec map func = function
        | Nil -> Nil
        | Cons (x,l) -> Cons ( func x , map func l);;
```

test_my_list.ml

```
let string_of_nat_list = string_of_list string_of_int in
let string_of_string_list = string_of_list (fun x -> x) in

let empty = Nil in
let one = Cons ("a", Nil) in
(*let lst = [1; 3; 6; 10; 15; 21; 28; 36; 45; 55];;*)
let lst = Cons(1, Cons(3, Cons(6, Cons(10, Cons(15, Cons(21, Cons(28, Cons(36, Cons
    (45, Cons(55,Nil))))))))) in

let match2 = function
  | None -> ""
  | Some l -> l in

let test_hd () =
  Printf.printf "Tte de %s : %s.\n" (string_of_string_list one) (match2 (hd one));
  let a = hd lst in match a with |None -> Printf.printf "Tte de %s : pas de tte.\n\n
      " (string_of_nat_list lst)
                            |Some l -> Printf.printf "Tte de %s : %d.\n\n" (
                                string_of_nat_list lst) l in

let test_tl () =
  Printf.printf "Queue de %s : %s.\n" (string_of_string_list one) (
      string_of_string_list (match (tl one) with |None -> Nil |Some l -> l));
  let a = tl lst in match a with |None -> Printf.printf "Queue de %s : pas de queue
      .\n\n" (string_of_nat_list lst)
                            |Some l -> Printf.printf "Queue de %s : %s.\n\n" (
                                string_of_nat_list lst) (string_of_nat_list l) in

let test_length () =
  Printf.printf "Taille de %s : %d.\n" (string_of_string_list one) (length one);
  Printf.printf "Taille de %s : %d.\n" (string_of_nat_list lst) (length lst);
  Printf.printf "Taille de %s : %d.\n\n" (string_of_string_list empty) (length empty
      ) in

let test_map () =
  Printf.printf "Map de (x -> xx) sur %s : %s.\n" (string_of_string_list one) (
      string_of_string_list (map (fun s -> s ^ s) one));
  Printf.printf "Map de (x -> 2x) sur %s : %s.\n" (string_of_nat_list lst) (
      string_of_nat_list (map (fun n -> 2 * n) lst));
  Printf.printf "Map de (x -> 2x) sur %s : %s.\n\n" (string_of_nat_list empty) (
      string_of_nat_list (map (fun n -> 2 * n) empty))

in test_hd();
test_tl();
test_length();
test_map();
```