```
let list = [1;2;3]

E4
let sum_cube l =
  let rec sum acc l =
    match l with
    | [] -> acc
    | x :: s -> sum (acc + x * x * x) s
  in sum 0 l;;

E5
let sum_cube l = List.fold_left (fun a b -> b * b*b + a) 0 l;;

E6
(*Je comprends pas la question*)

E7
(*Je comprends pas la question*)

type ft =
| Leaf of int
| Node of ft * int * int * ft;;

E8
let ft = Node(Node(Leaf(4), 1, 13, Leaf(9)), 2, 30,
    Node(Leaf(8), 1, 17, Node(Leaf(2), 1, 9, Leaf(7))));;
let t' = [|4;9;8;2;7|];;
E9
let rec create lo hi =
  if hi = lo+1 then Leaf(0)
  else let mid = (lo + hi) /2 in
  Node(create lo mid, mid - lo, 0, create mid hi);;
E10
let rec add indice valeur ft =
  match ft with
  | Leaf(v) -> Leaf(v+valeur)
  | Node(g, ng, sum, d) -> if indice < ng
                           then Node(add indice valeur g, ng+1, sum+valeur, d)
                           else Node(g, ng, sum+valeur, add indice valeur d);;

E11
let sum ft =
  match ft with
  | Leaf(v) -> v
  | Node(g, ng, sum, d) -> sum;;

E12
let rec prefix_sum i ft =
  match ft with
  | Leaf(v) -> 0
  | Node(g, ng, s, d) -> if i = ng then sum g
                           else if i < ng then prefix_sum i g
                           else sum g + prefix_sum (i - ng) d;;
E13
let between lo hi ft = prefix_sum hi ft - prefix_sum lo ft;;
```