(\* TME 4 \*)

(\* Representation des ensembles finis par des listes sans doublons \*)

(\* Q1 \*)

let rec is\_in (e: 'a) (l : 'a list) : bool =

match l with

[]->false

|x::xs'-> e=x || (is\_in e xs')

(\* Q2 \*)

let add\_elem (e: 'a) (l : 'a list) : 'a list =

if (is\_in e l) then l else e::l

(\* Q3 \*)

let rec is\_subset\_rec (l1: 'a list) (l2: 'a list) : bool =

match l1 with

[]->true

|x::xs-> if (is\_in x l2) then (is\_subset\_rec xs l2) else false

(\* Q4 \*)

let is\_subset (l1: 'a list) (l2: 'a list) : bool =

List.for\_all (fun x -> (is\_in x l2)) l1

(\* Q5 \*)

let eq\_set (l1: 'a list) (l2: 'a list) : bool =

if (is\_subset l1 l2 && is\_subset l2 l1) then true else false

(\* Q6 \*)

let rec intersection\_rec (l1: 'a list) (l2: 'a list) : 'a list =

match l1,l2 with

|(l1,[]) -> []

|([],l2) -> []

|(x::sl1,l2) -> if (is\_in x l2) then x::(intersection\_rec sl1 l2) else (intersection\_rec sl1 l2)

(\* Q7 \*)

let intersection (l1: 'a list) (l2: 'a list) : 'a list =

List.filter (fun x -> (is\_in x l2)) l1

(\* Q8 \*)

let rec union\_rec (l1: 'a list) (l2: 'a list) : 'a list =

match l1,l2 with

|([],l2) -> l2

|(x::sl1,l2) -> (union\_rec sl1 (add\_elem x l2))

(\* Q9 \*)

let union\_left (l1: 'a list) (l2: 'a list) : 'a list =

List.fold\_left (fun x -> fun y -> (add\_elem y x)) l1 l2

(\* Q10 \*)

let union\_right (l1: 'a list) (l2: 'a list) : 'a list =

List.fold\_right (fun x -> fun y -> (add\_elem x y)) l1 l2

(\* Q11 \*)

let make\_pairs (x:'a) (l: 'b list) : ('a \* 'b) list =

List.map (fun a -> (x,a)) l

(\* Q12 \*)

let rec product\_rec (l1: 'a list) (l2: 'b list) : ('a \* 'b) list =

match (l1,l2) with

|(l1,[]) -> []

|([],l2) -> []

|(x::ls,l2) -> (make\_pairs x l2)@(product\_rec ls l2)

(\* Q13 \*)

let product (l1: 'a list) (l2: 'b list) : ('a \* 'b) list =

List.flatten (List.map (fun x -> (make\_pairs x l2)) l1)

(\* Q14 \*)

let rec powerset\_rec (l:'a list) : 'a list list =

match l with

[] -> [[]]

|x::ls -> union\_rec (powerset\_rec ls) (List.map (fun y -> add\_elem x y) (powerset\_rec ls))

(\* Q15 \*)

let power (l:'a list) : ('a list) list =

List.fold\_left(fun z y ->z @(List.map(fun x-> y::x )z))[[]] l