(\* TP3 \*)

(\* Tri fusion \*)

(\* Q1 \*)

let rec merge (xs: 'a list) (ys: 'a list) : 'a list =

match (xs,ys) with

([],ys)->ys

|(xs,[])->xs

|(x::xs',y::ys') -> if x<y then x::(merge xs' ys) else y::(merge xs ys')

(\* Q2 \*)

let rec split (xs: 'a list) : 'a list \* 'a list =

match xs with

[]->([],[])

|x1::[]->(x1::[],[])

|x1::x2::xs'-> let l1,l2 = (split xs') in x1::l1,x2::l2

(\* Q3 \*)

let rec merge\_sort (xs: 'a list) : 'a list =

match xs with

[] -> xs

|x::[] -> xs

|xs -> let l1, l2 = split(xs) in merge (merge\_sort l1) (merge\_sort l2)

(\* D'autres ordres \*)

(\* Q1 \*)

let rec merge\_gen (cmp: 'a -> 'a -> bool) (xs: 'a list) (ys: 'a list) : 'a list =

match (xs, ys) with

([], ys)-> ys

|(xs, []) -> xs

|x::xxs, y::yys -> if (cmp x y) then x::(merge\_gen cmp xxs (y::yys))

else y::(merge\_gen cmp (x::xxs) yys)

(\* Q2 \*)

let rec merge\_sort\_gen (cmp: 'a -> 'a -> bool) (xs: 'a list) : 'a list =

match xs with

[]->xs

|x::[]->xs

|xs -> let l1, l2 = split(xs) in merge\_gen cmp (merge\_sort\_gen cmp l1) (merge\_sort\_gen cmp l2)

(\* Q3 \*)

let sort\_weight (xs: (int \* int) list) : (int \* int) list =

let tri\_double((x1,x2) : (int\*int)) ((y1,y2) : (int\*int)):

bool = x1 + x2 <= y1+y2 in

(merge\_sort\_gen tri\_double xs)