1. *Tri rapide.*

```
let rec tri_rapide  = function
  |  []    -> []
  |  [e]   -> [e]
  |  e::r  -> let l1,l2 = partition  e r in
              (tri_rapide l1)@(e::(tri_rapide l2));;

let rec partition (e:int)  = function
| [] -> ([], [])
| d::r -> let l1,l2 = partition e r in
if (d < e) then (d::l1,l2) else (l1,d::l2);;
```

2. *Le tri par sélection.*

```
let rec minimum_et_reste = function
    | [x] ->(x,[])
    | x1::r1 -> let (m2,l2) = (minimum_et_reste r1) in
                if x1<m2 then (x1,m2::l2) else (m2,x1::l2);;

let rec tri_selection = function
    | [] -> []
    | l  -> let (m,r) = (minimum_et_reste l) in
                         m::(tri_selection r);;
```

3. *Le tri par insertion*

```
let rec insere element = function
    | [] -> [element]
    | x::reste -> if element <= x
                  then element::x::reste
                  else x::(insere element reste);;

let rec tri_insertion = function
    | [] -> []
    | x::reste -> insere x (tri_insertion reste);;
```

4. *Le tri bulle.*

```
let rec une_passe  = function
  | [(x:int)] -> false,[x]
  | x::reste  -> let bool,res = (une_passe reste) in
                   if x<=(hd res)
                   then bool,x::res
                   else true,(hd res)::x::(tl res);;

let rec tri_bulle  = function
  | [] -> []
  | l -> let (modifiee, liste) = une_passe l in
           if modifiee
           then (hd liste)::(tri_bulle (tl liste))
           else liste;;
```

5. *Le tri fusion.*

```
let rec divise = function
    | [] -> ([],[])
    | [e] -> ([e],[])
    | a::b::r -> let (m1,m2) = divise r in
                 (a::m1,b::m2);;

let rec fusion = fun
    | l [] -> l
    | [] l -> l
    | (a::r as l1) (b::s as l2) -> if a<b
                                   then a::(fusion r l2)
                                   else b::(fusion l1 s);;

let rec tri_fusion = fun
    | [] -> []
    | [e] -> [e]
    | l -> let (m1,m2) = divise l in
           fusion (tri_fusion m1) (tri_fusion m2);;
```