## 課題 3.1

```
(* リストに値x が何個含まれているかを返す関数 *)
let rec num_of (lst, x) =
  match lst with
    [] -> 0
  | n :: rest ->
       if n = x then 1 + num_of (rest, x)
       else num_of (rest, x);;

let num_of' (lst, x) =
  let rec num (lst, acc) =
    match lst with
      [] -> acc
    | n :: rest ->
      if n = x then num(rest, acc + 1)
      else num (rest, acc) in
  num (lst, 0);;
```

実行結果

```
# num_of ([4; 3; 2; 3; 4; 3], 3);;
- : int = 3
# num_of' ([4; 3; 2; 3; 4; 3], 3);;
- : int = 3
# num_of ([4; 3; 2; 3; 4; 3], 4);;
- : int = 2
# num_of' ([4; 3; 2; 3; 4; 3], 4);;
- : int = 2
```

## 課題 3.2

```
(* 整数リストが与えられ負の要素の和と正の要素の和の組を求める *)
let sum_pair' lst =
  let rec sums (lst, ltz, gtz) =
    match lst with
      [] -> (ltz, gtz)
    | x :: rest ->
      if x > 0 then sums (rest, ltz, x + gtz) else sums (rest, x + ltz, gtz) in
  sums (lst, 0, 0);;
```

実行結果

```
# sum_pair' [-2; 0; 3; -1; 2; 1];;
- : int * int = (-3, 6)
# sum_pair' [1; 2; 3; -4; 5; 6];;
- : int * int = (-4, 17)
# sum_pair' [-2; 0; 3; 1; 2; 1];;
- : int * int = (-2, 7)
```

## 課題 3.3

```
(* マージソート *)
(* リストの偶数番目と奇数番目の要素に分割する *)
let rec split_even_odd lst =
  match lst with
    [] -> ([], [])
  | [m] -> ([m], [])
  | m :: n :: rest ->
     let (even, odd) = split_even_odd rest in
     (m :: even, n :: odd);;

let rec merge (xl, yl) =
  match (xl, yl) with
    ([], _) -> yl
  | (_, []) -> xl
  | (x::restx, y::resty) ->
     if x < y then x::merge (restx, yl)
     else y::merge (xl, resty);;

let rec msort lst =
  let (lst1, lst2) = split_even_odd lst in
  match (lst1, lst2) with
    ([], []) -> []
  | (l1 :: [], []) -> merge (lst1, lst2)
  | ([], l2 :: []) -> merge (lst1, lst2)
  | (l1 :: rest1, []) -> merge ((msort lst1), lst2)
  | ([], l2 :: rest2) -> merge (lst1, (msort lst2))
  | (l1 :: rest1, l2 :: rest2) -> merge ((msort lst1), (msort lst2));;
```

実行結果

```
# msort [4; 5; 2; 1];;
- : int list = [1; 2; 4; 5]
# msort [10; 9; 8; 7; 6; 5; 4; 3; 2; 1];;
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
```

## 課題 3.4

```
(* フィボナッチ数列の改善 *)
let fib n =
  let rec iterfib (i, acc1, acc2) =
    if i = n then acc1
    else iterfib (i+1, acc1+acc2, acc1) in
  iterfib (0, 0, 1);;

let rec fib2 n =
  if n = 0 then (0, 1)
  else let (a, b) = fib2 (n-1) in (b, a + b);;
```

実行結果

```
# fib 10;;
- : int = 55
# fib 20;;
- : int = 6765
# fib 30;;
- : int = 832040
# fib 40;;
- : int = 102334155
# fib2 10;;
- : int * int = (55, 89)
# fib2 11;;
- : int * int = (89, 144)
# fib2 12;;
- : int * int = (144, 233)
```