ソフトウェア技法　第 5 回　201311367　佐藤良祐 2014 年 11 月 12 日

課題 5.1

```
(* 関数f, 整数mに対して i=0からmの f(i)の総和を求める *)
let summation f m =
  let rec sums f m =
    if m = 0 then f m else f m + sums f (m -1) in
  sums f m;;

(* 上の関数を用いて関数g, 整数m, nを計算する関数をかけ *)
let summation2 f (m, n) =
  summation (fun x -> summation (fun y -> f (x, y)) m) n;;
```

実行結果

```
# summation (fun x -> x) 3;;
- : int = 6
# summation (fun x -> x * x) 3;;
- : int = 14
# summation2 (fun (x, y) -> x + y) (2, 2);;
- : int = 18
# summation2 (fun (x, y) -> x * y) (2, 2);;
- : int = 9
```

課題 5.2

```
(* リストを集合とし考えた時に集合の交わりを計算する *)
let rec inter l1 l2 =
  match l1 with
    [] -> []
  | x1::rest ->
     let rec filter pred l =
       match l with
         [] -> []
       | x::rest ->
          if pred x then x::filter pred rest
          else filter pred rest in
     filter (fun x -> x = x1) l2 @ inter rest l2;;

(* 集合の全ての要素をある値と対にする関数 *)
let rec pair v lst =
  match lst with
    [] -> []
  | x::rest ->
     let rec map f l =
       match l with
         [] -> []
       | x::rest -> f x :: map f rest in
     map (fun x -> (v, x)) lst;;

(* 2つの集合がリストで与えられたとき直積集合を計算する *)
let rec prod lst1 lst2 =
  match lst1 with
    [] -> []
  | x::rest -> pair x lst2 @ prod rest lst2;;
```

実行結果

```
# inter [3; 1; 2;] [2; 3];;
- : int list = [3; 2]
# inter [4; 5; 1; 6; 3] [2; 4; 3];;
- : int list = [4; 3]
# pair 1 ["A"; "B"; "C"];;
- : (int * string) list = [(1, "A"); (1, "B"); (1, "C")]
- : (int * string) list =
[(1, "A"); (1, "B"); (2, "A"); (2, "B"); (3, "A"); (3, "B")]
```

課題 5.3

```
(* クイックソートのプログラムの要素の順序を決める関数を引数として取るようにする *)
let rec qsort le l =
  match l with
    [] -> []
  | b::rest ->
     let rec split le l =
       match l with
         [] -> ([], [])
       | x::rest ->
          let (l1, l2) = split le rest in
          if le x b then (x::l1, l2) else (l1, x::l2) in
     let (l1, l2) = split le rest in
     qsort le l1 @ (b::qsort le l2);;
```

実行結果

```
qsort (fun x y -> (x <= y)) [2; 1; 2; 3];;
- : int list = [1; 2; 2; 3]
# qsort (fun x y -> x >= y) [4.0; 3.0; 7.0];;
- : float list = [7.; 4.; 3.]
```

課題 5.4

```
(* 二分木のデータ型 *)
type 'a btree =
  Lf
  | Br of 'a * 'a btree * 'a btree;;


(* 二分探索木に新しい要素を挿入する *)
let rec add x tr =
  match tr with
    Lf -> Br (x, Lf, Lf)
  | Br (v, left, right) -> if v > x then Br (v, add x left, right) else Br (v,
      left, add x right);;


(* 二分探索木の中の最小の要素を返す関数 *)
let rec min_elt tr =
  match tr with
```

```
    Lf -> raise (Failure "min_elt")
  | Br (v, left, right) -> if left = Lf then v else min_elt left;;


(* 二分探索木の要素を削除する *)
let rec remove n tr =
  match tr with
    Lf -> Lf
  | Br (v, left, right) ->
     if n < v then Br (v, remove n left, right) (* 左の子へ進む *)
     else if n > v then Br (v, left, remove n right) (* 右の子へ進む *)
     else
       if left = Lf then right
       else if right = Lf then left
       else let min = min_elt right in
           let rec remove_min ts =
             match ts with
               Lf -> Lf
             | Br (v, Lf, right) -> right
             | Br (v, left, right) -> Br (v, remove_min left, right) in
           Br (min, left, remove_min right);;
```

実行結果

```
let tree = Lf;;
val tree : 'a btree = Lf
# let tree = add 8 tree;;
val tree : int btree = Br (8, Lf, Lf)
# let tree = add 3 tree;;
val tree : int btree = Br (8, Br (3, Lf, Lf), Lf)
# let tree = add 10 tree;;
val tree : int btree = Br (8, Br (3, Lf, Lf), Br (10, Lf, Lf))
# let tree = add 1 tree;;
val tree : int btree = Br (8, Br (3, Br (1, Lf, Lf), Lf), Br (10, Lf, Lf))
# let tree = add 6 tree;;
val tree : int btree =
  Br (8, Br (3, Br (1, Lf, Lf), Br (6, Lf, Lf)), Br (10, Lf, Lf))
# let tree = add 14 tree;;
val tree : int btree =
  Br (8, Br (3, Br (1, Lf, Lf), Br (6, Lf, Lf)),
   Br (10, Lf, Br (14, Lf, Lf)))
# min_elt tree;;
- : int = 1
# remove 10 tree;;
- : int btree =
```