ソフトウェア技法　第 4 回　201311367　佐藤良祐 2014 年 10 月 22 日

課題 4.1

```
(* 円,正方形,長方形の図形を定義する *)
type figure =
  | Circle of float
  | Square of float
  | Rectangle of float * float;;

let area figure =
  match figure with
  | Circle (r) -> r *. r *. (4.0 *. atan 1.0)
  | Square (x) -> x *. x
  | Rectangle(x, y) -> x *. y;;
```

実行結果

```
# Circle 2.0;;
- : figure = Circle 2.
# Square 4.0;;
- : figure = Square 4.
# Rectangle (2., 4.);;
- : figure = Rectangle (2., 4.)
# area (Circle 1.);;
- : float = 3.14159265358979312
# area (Square 4.);;
- : float = 16.
# area (Rectangle (2., 4.));;
- : float = 8.
```

課題 4.2

```
(* 木の定義 *)
type 'a tree =
  Lf
  | Br of 'a * 'a tree * 'a tree;;

(* 木の例 *)
let tree2= Br ("A", Br ("B", Br ("C", Lf, Lf),
                    Br ("D", Br ("E", Lf, Lf),
                       Lf)),
              Br ("F", Lf, Lf));;
(* 木の深さを計算する *)
let rec depth t =
  match t with
    Lf -> 0
  | Br (v, t1, t2) -> 1 + max (depth t1) (depth t2)

(* 深さがn ですべてのノードのラベルが x の完全 2 分木を作る関数 *)
let rec comptree (n, x) =
  match n with
    0 -> Lf
  | _ -> Br (x, comptree(n-1, x), comptree(n-1, x));;
```

実行結果

```
# depth tree2;;
- : int = 4
# comptree(3, "A");;
- : string tree =
Br ("A", Br ("A", Br ("A", Lf, Lf), Br ("A", Lf, Lf)),
 Br ("A", Br ("A", Lf, Lf), Br ("A", Lf, Lf)))
```

課題 4.3

```
(* 木の定義 *)
type 'a tree =
  Lf
  | Br of 'a * 'a tree * 'a tree;;

(* 木の例 *)
let tree2= Br ("A", Br ("B", Br ("C", Lf, Lf),
                     Br ("D", Br ("E", Lf, Lf),
                        Lf)),
             Br ("F", Lf, Lf));;

(* 木を中順で走査しリストとして返す *)
let rec inorder t =
  match t with
    Lf -> []
  | Br (v, t1, t2) -> inorder t1 @ [v] @ inorder t2;;

(* 木を後順で走査しリストとして返す *)
let rec postorder t =
  match t with
    Lf -> []
  | Br (v, t1, t2) -> postorder t1 @ postorder t2 @ [v];;
```

実行結果

```
# inorder tree2;;
- : string list = ["C"; "B"; "E"; "D"; "A"; "F"]
# postorder tree2;;
- : string list = ["C"; "E"; "D"; "B"; "F"; "A"]
```

課題 4.4

```
(* 有限分岐の木の定義 *)
type 'a ftree = FBr of 'a * 'a ftree list;;

(* 有限分岐の木の例 *)
let ftree = (FBr (1, [FBr (2,[]) ;FBr (3, [FBr (4, [])]); FBr (5, [])]));;

(* 有限分岐の木に対して木の深さを返す *)
let rec fdepth t =
  match t with
    FBr (v, ts) -> fdepth_ts ts
  and fdepth_ts ts =
    match ts with
      [] -> 0
    | t::ts' -> fdepth t + fdepth_ts ts' + 1;;

(* 有限分岐の木を先順で走査する関数 *)
let rec fpreorder t =
  match t with
    FBr (v, ts) -> [v] @ fpreorder_ts ts
and fpreorder_ts ts =
  match ts with
    [] -> []
  | x::rest -> fpreorder x @ fpreorder_ts rest;;
```

実行結果

```
# fdepth ftree;;
- : int = 4
# fpreorder ftree;;
- : int list = [1; 2; 3; 4; 5]
```