課題1

```
type figure = Circle of float | Square of float | Rectangle of float * float;;
Circle 3.;;
Rectangle (5., 6.);;

let area x =
        match x with
                Circle r -> r *. r *. 3.14
                | Square a -> a *. a
                | Rectangle (h, w) -> h *. w;;
area (Circle 2.);;
area (Rectangle (2., 4.));;
```

実行結果

```
type figure = Circle of float | Square of float | Rectangle of float * float
- : figure = Circle 3.
- : figure = Rectangle (5., 6.)
val area : figure -> float = <fun>
- : float = 12.56
- : float = 8.
```

課題2

```
type 'a tree = Lf | Br of 'a * 'a tree * 'a tree;;
let tree1 = Br (1, Br (2, Lf, Lf), Br (3, Lf, Lf));;
let tree2 = Br("A", Br("B", Br("C", Lf, Lf), Br("D", Br("E", Lf, Lf), Lf)), Br("F", Lf, Lf));;

let rec depth t =
        match t with
                Lf -> 0
                | Br (_, t1, t2) -> 1 + max (depth t1) (depth t2);;
depth tree1;;
depth tree2;;

let rec comptree (x, n) =
        match n with
                0 -> Lf
                | num -> Br(x, comptree(x, num-1), comptree(x, num-1));;
comptree ("A", 0);;
comptree ("A", 1);;
comptree ("A", 2);;
comptree ("A", 3);;
```

実行結果

```
type 'a tree = Lf | Br of 'a * 'a tree * 'a tree
val tree1 : int tree = Br (1, Br (2, Lf, Lf), Br (3, Lf, Lf))
val tree2 : string tree =
  Br ("A", Br ("B", Br ("C", Lf, Lf), Br ("D", Br ("E", Lf, Lf), Lf)), Br ("F", Lf, Lf))
val depth : 'a tree -> int = <fun>
- : int = 2
- : int = 4
val comptree : 'a * int -> 'a tree = <fun>
- : string tree = Lf
- : string tree = Br ("A", Lf, Lf)
- : string tree = Br ("A", Br ("A", Lf, Lf), Br ("A", Lf, Lf))
- : 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3

```
type 'a tree = Lf | Br of 'a * 'a tree * 'a tree;;
let tree1 = Br (1, Br (2, Lf, Lf), Br (3, Lf, Lf));;
let tree2 = Br("A", Br("B", Br("C", Lf, Lf), Br("D", Br("E", Lf, Lf), Lf)), Br("F", Lf, Lf));;

let rec inorder t =
        match t with
                Lf -> []
                | Br (v, t1, t2) -> inorder t1 @ [v] @ inorder t2;;
let rec postorder t =
        match t with
                Lf -> []
                | Br (v, t1, t2) -> postorder t1 @ postorder t2 @ [v];;
inorder tree1;;
inorder tree2;;
postorder tree1;;
postorder tree2;;
```

実行結果

```
type 'a tree = Lf | Br of 'a * 'a tree * 'a tree
val tree1 : int tree = Br (1, Br (2, Lf, Lf), Br (3, Lf, Lf))
val tree2 : string tree =
  Br ("A", Br ("B", Br ("C", Lf, Lf), Br ("D", Br ("E", Lf, Lf), Lf)), Br ("F", Lf, Lf))
val inorder : 'a tree -> 'a list = <fun>
val postorder : 'a tree -> 'a list = <fun>
- : int list = [2; 1; 3]
- : string list = ["C"; "B"; "E"; "D"; "A"; "F"]
- : int list = [2; 3; 1]
- : string list = ["C"; "E"; "D"; "B"; "F"; "A"]
```

課題4

```
type 'a ftree = FBr of 'a * 'a ftree list;;
let ftree1 = FBr(1, [FBr(2, []); FBr(3, [FBr(4, [])])]);;
let ftree2 = FBr(1, [FBr(2, [FBr(3, []); FBr(4, [])]); FBr(5, [])]);;

let rec fdepth t =
        match t with
                FBr (v, ts) -> fdepth_ts ts + 1
and fdepth_ts ts =
        match ts with
                [] -> 0
                | t::rest -> max (fdepth t) (fdepth_ts rest);;

let rec fpreorder t =
        match t with
                FBr (v, ts) -> [v] @ fpreorder_ts ts
and fpreorder_ts ts =
        match ts with
                [] -> []
                | t::rest -> fpreorder t @ fpreorder_ts rest;;

fdepth ftree1;;
fdepth ftree2;;
fpreorder ftree1;;
fpreorder ftree2;;
```

実行結果

```
type 'a ftree = FBr of 'a * 'a ftree list
val ftree1 : int ftree = FBr (1, [FBr (2, []); FBr (3, [FBr (4, [])])])
val ftree2 : int ftree = FBr (1, [FBr (2, [FBr (3, []); FBr (4, [])]); FBr (5, [])])
val fdepth : 'a ftree -> int = <fun>
val fdepth_ts : 'a ftree list -> int = <fun>
val fpreorder : 'a ftree -> 'a list = <fun>
val fpreorder_ts : 'a ftree list -> 'a list = <fun>
- : int = 3
- : int = 3
- : int list = [1; 2; 3; 4]
- : int list = [1; 2; 3; 4; 5]
```