Homework 8
SNU 4910.210 Fall 2008

Kwangkeun Yi

due: 12/11(Thu) 24:00

Exercise 1 "종이 벽지 디자인 ML"

벽지 무늬의 전체구조는 대게가 작은 기본 무늬들의 반복입니다. 기본 무늬는 정 사각형에 그려져있다고 합시다. 무늬를 디자인하는 작업은 기본 정사각형들을 4개로 짜집기해서 큰 정사각형의 모양을 만들고, 이것들 4개로 다시 보다 큰 정사격형을 만들고, 등등. 되었다 싶으면 디자인된 정 사각형들을 반복해서 창살에 짜넣는 방법을 취하겠지요.

그 작업 과정을 functor로 표현할 수 있는데, 다음과 같습니다. 코드를 잘 읽어보고, 코드에서 "..." 부분을 메꾸어 보세요.

```
(* - four designs (A,B,C,D) for basic box
   - four boxes are glued together to become a bigger box
*)
type design = TURTLE | WAVE | DRAGON | SNU (* four design patterns *)
type orientation = NW | NE | SE | SW
type box = BOX of orientation * design | GLUED of box * box * box * box
signature FRAME =
  sig
   val box: box
   val rotate: box -> box           (* rotate box M to 3 to W to E *)
   val pp: box -> int * int -> unit (* pretty printer *)
  end
functor BasicFrame(Design: sig val pattern: design end): FRAME =
  struct
    exception NON_BASIC_BOX
    val box = BOX (NW, Design.pattern)    (* a box is defined *)
```

```
    val rotate = ...
    fun pp (BOX(NW,x)) center = ()
      | pp (BOX(NE,x)) center = ()
      | pp (BOX(SE,x)) center = ()
      | pp (BOX(SW,x)) center = ()
      | pp _ _ = raise NON_BASIC_BOX
  end
functor Rotate(Box: FRAME): FRAME =
  struct
   val box = ...
   val rotate = ...
   val pp = ...
  end
functor Glue(structure Nw: FRAME
             structure Ne: FRAME
             structure Se: FRAME
             structure Sw: FRAME): FRAME =
  struct
   ...
  end
```

자, 위의 코드를 이용한 다음의 디자인과정을 보세요:

```
structure A = BasicFrame(struct val pattern = TURTLE end)
structure B = BasicFrame(struct val pattern = WAVE end)
structure A' = Rotate(A)   (* another block rotated by 90 degrees clockwise *)
structure A'' = Rotate(A') (* another block rotated by 90 degrees clockwise *)
structure B' = Rotate(B)   (* another block rotated by 90 degrees clockwise *)
structure B'' = Rotate(B') (* another block rotated by 90 degrees clockwise *)
structure BigA = Glue(structure Nw = A
                      structure Ne = B
                      structure Se = A'
                      structure Sw = B')
structure BigB = Glue(structure Nw = A
                      structure Ne = A'
                      structure Se = B
                      structure Sw = B')
```

```
structure BigA' = Rotate(BigA)
structure BigB' = Rotate(BigB)
structure M = Glue(structure Nw = BigA
                   structure Ne = BigB
                   structure Se = BigA'
                   structure Sw = BigB')
val bluePrint = M.pp M.box
```

□

Exercise 2 "집합 모듈 함수"

집합 모듈을 만들어 주는 아래의 모듈 함수들(SetFun, ProductSetFun, PowerSetFun)을 정의해봅시다. "..." 부분을 메꾸면 됩니다. 만들어 지는 모든 집합 모듈들은 아래의 SET 시그너쳐를 만족해야 합니다.

```
signature SET =
      sig
       type t
       type 'a set
       val empty : t set
       val equal : t * t -> bool
       val member : t * t set -> bool
       val add : t * t set -> t set
       val cup : t set * t set -> t set
       val cap : t set * t set -> t set
       val minus : t set * t set -> t set
      end

functor SetFun(S: sig type t val equal: t * t -> bool end) =
  struct
   type t = S.t
   type 'a set = Empty | Set of 'a * 'a set
   ...
  end

functor ProductSetFun(A:SET, B:SET) =
  struct
```

```
    type t = A.t * B.t
    type 'a set = Empty | Set of 'a * 'a set
    ...
  end

functor PowerSetFun(A:SET) =
  struct
    type t = A.t A.set
    type 'a set = Empty | Set of 'a * 'a set
    ...
  end
```

그래서, 아래와 같은 프로그래밍이 가능해야 한다.

```
structure IntSet = SetFun(struct
                            type t = int
                            fun equal(x,y) = x=y
                          end)

val a = IntSet.add(1,IntSet.empty)    (* {1} *)
val b = IntSet.add(2,a)               (* {1,2} *)
val c = IntSet.cup(a,b)               (* {1}U{1,2} *)

structure StringSet = SetFun(struct
                                    type t = string
                                    fun equal(x,y) = x=y
                               end)
val d = StringSet.add(''a'', StringSet.empty)     (* {''a''} *)
val e = StringSet.minus(d,d)                      (* {} *)

structure A = ProductSetFun(IntSet, StringSet)

val f = A.add((1, ``a''), A.empty)  (* {<1,''a''>} *)
val g = A.add((2, ``c''), f)        (* {<2,''c''>, <1,''a''>} *)
val h = A.cup(f,g)                  (* {<2,''c''>, <1,''a''>} *)
```

```
structure B = PowerSetFun(IntSet)

val i = B.add(a, B.empty)        (* {{1}} *)
val j = B.add(b, i)              (* {{1, 2}, {1}} *)
val k = B.cap(i,j)               (* {{1}} *)
```

□