Homework 3
SNU 4910.210 Fall 2010

Kwangkeun Yi

due: 10/06 24:00

이번 숙제의 목적은:

- 타입을 따지면서 프로그램하는 것을 익힌다.
- 데이터의 속 구현을 프로그램 해 본다.
- 데이터의 속 구현을 여러가지 방식으로 프로그램 해 본다.
- 속 구현을 보지말고 인터페이스만 알고 프로그램하는 것을 익힌다.
- 속 구현이 여럿인 경우도 인터페이스만 알고 프로그램하는 것을 익힌다.

Exercise 1 "미로 만들기"

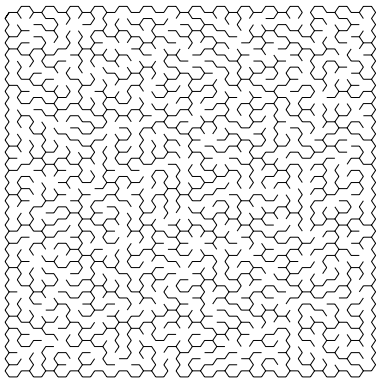
$n \times m$ 개의 정육각형 방을 가진 모눈종이를 생각하자. 미로의 정의는: 입구와 출구가 맨 윗줄의 방과 맨 아랫줄의 방에 각각 하나이고, 입구에서 출구까지의 통로가 유일한것을 뜻한다.

미로를 만드는 함수

$$\texttt{mazeGen} : \mathit{int} \times \mathit{int} \rightarrow \mathit{maze}$$

를 작성하라. 즉, (`mazeGen` n m) 은 양수 n과 m을 받아 $n \times m$ 육각모눈종이에 미로를 만들어 준다.

육각미로의 한 예:



다음의 함수들을 사용해서 정의하라.

`init-maze` : $int \times int \to maze$
`open-e` : $int \times int \times maze \to maze$
`open-w` : $int \times int \times maze \to maze$
`open-se` : $int \times int \times maze \to maze$
`open-sw` : $int \times int \times maze \to maze$
`open-ne` : $int \times int \times maze \to maze$
`open-nw` : $int \times int \times maze \to maze$
`maze-pp` : $maze \to void$

(`init-maze` n m)는 미로인데 모든 방들의 여섯개 벽이 모두 막혀있다. 모든 $n \times m$개의 방들은 좌표로 구분되고, 좌표는 $(0,0)$부터 $(n-1, m-1)$까지가 된다. (`open-`d n m M)는 미로 M에서 방 (n,m)의 d-방향 벽을 튼 미로를 만든다. (`maze-pp` M)는 미로 M을 이쁘게 그려준다.

참고로, 미로를 만들때의 목표는 될 수 있으면 어렵게 만들자는 것이다. 간단히는 다음과 같이 만들 수 있다. 일단 입구에서 출구까지 방들의 벽을 터서 해답 통로를 만들고, 나머지 방들의 벽을 적당히 터 주어서 혼동스러운 모습을 띄도록 한다.

하지만, 이 방법은 비교적 찾기 쉬운 미로를 만든다. 해답 통로가 눈에 띄기 쉽고, 해답이 아닌 통로의 길이가 상대적으로 짧게되기 쉽다. 더 좋은 방법은,

임의의 방들의 벽을 임의로 터가는 것이다. 언제까지? 적어도 입구방과 출구방이 연결될 때 까지. □

Exercise 2 "종이 벽지 디자인"

벽지 무늬의 전체구조는 대게가 같은 무늬들의 반복이다. 기본 무늬는 검거나 흰 정사각형이다. 무늬를 디자인하는 작업은 기본 정사각형들 4개를 연결해서 4배 큰 정사각형 무늬를 만들고, 이것들 4개를 다시 연결해서 4배 더 큰 정사각형을 만들고, 등등. 되었다 싶으면 디자인된 정 사각형들을 반복해서 종이에 짜넣는 방법을 취한다.

이러한 무늬 데이타의 속 내용을 감추는 다음의 함수들을 정의하라.

`black` : 무늬
`white` : 무늬
`glue` : 무늬 × 무늬 × 무늬 × 무늬 → 무늬
`rotate` : 무늬 → 무늬
`neighbor` : 위치 × 무늬 → int
`pprint` : 무늬 → void

각 함수들이 하는 일은 다음과 같다:

- `black`: 기본크기의 검은 정사각형 무늬.
- `white`: 기본크기의 흰 정사각형 무늬.
- `glue`: 같은 크키의 정사각형 무늬 4개를 NW, NE, SE, SW방향의 순서로 받아서 그 위치에 놓고 연결한 4배 크기의 정사각형 무늬를 만든다.
- `rotate`: 정사각형 무늬를 받아서 90도 시계방향으로 돌려진 무늬를 만든다.
- `neighbor`: 주어진 위치의 기본 정사각형의 주변에 있는 최대 8개의 정사각형중 검은 정사각형의 갯수. 기본 정사각형의 위치는 전체 정사각형에서 부터 시작해서 계속 4등분 해 가면서 그 정사각형이 포함된 구역의 번호들의 리스트이다. NW 구역은 0, NE 구역은 1, SE 구역은 2, SW 구역은 3이다. 무늬가 기본 정사각형 하나일 때는 그 정사각형의 위치는 빈 리스트가 된다. 예를 들어, 위치가 (3 3) 인 정사각형은 16개의 기본 정사각형으로 구성된 정사각형 판에서 가장 왼쪽 아래의 기본 정사각형을 말한다. 한 무늬가 가지는 기본 정사각형의 갯수는 4^i개 이고, 기본 정사각형의 위치를 표현하는 리스트의 길이는 항상 i가 된다. 이 조건을 만족할 때에만 `neighbor`가 정의된다.

- `pprint`: 정사각형 무늬를 화면에 그려준다.

예를 들어서 다음과 같이 벽지무늬들을 만들어서 프린트할 수 있겠다 (어떤 무늬가 프린트될까?)

```
(define B black)
(define W white)
(define Basic (glue B B B W))
(define (turn pattern i)
  (if (<= i 0) pattern else (turn (rotate pattern) (- i 1))))
(define Compound (glue Basic (turn Basic 1) (turn Basic 2) (turn Basic 3)))
```

위와 같은 프로그램을 고안하는 데, 무늬를 구현하는 방법이 두가지가 있다:

- 정사각형 무늬안에 있는 기본 정사각형들의 가로줄(row)의 리스트로 표현하는 방법. 예를 들어, 위의 예에서 `Basic`은 `((B B) (W B))`로, `Compound`는 `((B B W B) (W B B B) (B B B W) (B W B B))`로 표현되겠다.
- 잎새에 기본 정사각형이 매달린, 모든 가지가 4갈래로 갈라지는 트리 구조로 표현하는 방법.

이 두 가지 구현 방안을 구현하라. 이 두 가지 구현 방안을 모두 가지고 위의 여섯가지 함수를 정의하라. 이 때 데이타의 표현방식에 맞는 적절하게 두 가지 방식이 모두 사용되도록 정의한다.

배열로 구현하는 경우, 드러나는(인터페이스) 함수들:

`glue-array-from-tree` :	무늬 × 무늬 × 무늬 × 무늬 → 무늬
`glue-array-from-array` :	무늬 × 무늬 × 무늬 × 무늬 → 무늬
`rotate-array` :	무늬 → 무늬
`neighbor-array` :	위치 × 무늬 → int
`pprint-array` :	무늬 → void
`is-array?` :	무늬 → bool

트리로 구현하는 경우, 드러나는 함수들:

`glue-tree-from-tree` :	무늬 × 무늬 × 무늬 × 무늬 → 무늬
`glue-tree-from-array` :	무늬 × 무늬 × 무늬 × 무늬 → 무늬
`rotate-tree` :	무늬 → 무늬
`neighbor-tree` :	위치 × 무늬 → int
`pprint-tree` :	무늬 → void
`is-tree?` :	무늬 → bool

□

Exercise 3 "벽지 아가씨 심사위원"

벽지 무늬를 다루는 함수들에 다음의 함수를 추가로 정의하고:

$$\texttt{equal} : \text{무늬} \times \text{무늬} \to \text{bool}$$
$$\texttt{size} : \text{무늬} \to \text{int}$$

equal은 두 무늬가 같은지를 판별하고, size는 기본 정사각형의 갯수가 4^i일 때 i를 내놓는다. equal이 받아들이는 두개의 무늬들은 다르게 표현된 것들일 수 있다.

그렇게 드러난 함수들을 이용해서 함수 beautiful을 정의하라.

$$\texttt{beautiful} : \text{무늬} \to \text{bool}$$

함수 beautiful은 벽지 무늬가 중앙점을 기준으로 대칭이거나, 대칭이지 않다면 모든 정사각형의 이웃한 검은 정사각형들의 갯수가 1개보다 많고 6개보다 작을 때이다. □