Homework 3
SNU 4910.210 Fall 2008

Kwangkeun Yi

due: 10/10 24:00

Exercise 1 "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 속내용을 감추면서(data abstraction) 구현해보자. 위의 여섯가지 함수를 정의할 때 데이타의 표현방식에 맞는 적절한 함수들을 사용하도록 정의한다.

배열로 구현하는 경우, 드러나는 함수들:

$$\begin{aligned}
\texttt{glue-array-from-tree}: &\quad \text{무늬} \times \text{무늬} \times \text{무늬} \times \text{무늬} \to \text{무늬} \\
\texttt{glue-array-from-array}: &\quad \text{무늬} \times \text{무늬} \times \text{무늬} \times \text{무늬} \to \text{무늬} \\
\texttt{rotate-array}: &\quad \text{무늬} \to \text{무늬} \\
\texttt{neighbor-array}: &\quad \text{위치} \times \text{무늬} \to \text{int} \\
\texttt{pprint-array}: &\quad \text{무늬} \to \text{void} \\
\texttt{is-array?}: &\quad \text{무늬} \to \text{bool}
\end{aligned}$$

트리로 구현하는 경우, 드러나는 함수들:

$$\begin{aligned}
\texttt{glue-tree-from-tree}: &\quad \text{무늬} \times \text{무늬} \times \text{무늬} \times \text{무늬} \to \text{무늬} \\
\texttt{glue-tree-from-array}: &\quad \text{무늬} \times \text{무늬} \times \text{무늬} \times \text{무늬} \to \text{무늬} \\
\texttt{rotate-tree}: &\quad \text{무늬} \to \text{무늬} \\
\texttt{neighbor-tree}: &\quad \text{위치} \times \text{무늬} \to \text{int} \\
\texttt{pprint-tree}: &\quad \text{무늬} \to \text{void} \\
\texttt{is-tree?}: &\quad \text{무늬} \to \text{bool}
\end{aligned}$$

Exercise 2 "벽지 아가씨 심사위원"

벽지 무늬를 다루는 함수들에 다음의 함수를 추가로 정의하고:

$$\begin{aligned}
&\texttt{equal} : \text{무늬} \times \text{무늬} \to \text{bool} \\
&\texttt{size} : \text{무늬} \to \text{int}
\end{aligned}$$

`equal`은 두 무늬가 같은지를 판별하고, `size`는 기본 정사각형의 갯수가 4^i일 때 i를 내놓는다. `equal`이 받아들이는 두개의 무늬들은 다르게 표현된 것들일 수 있다.

그렇게 드러난 함수들을 이용해서 함수 `beautiful`을 정의하라.

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

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

Exercise 3 "어울리지 않아, 정품"

이번 숙제는 올바른 프로그램 인지를 확인해 보기가 쉽지 않은 문제를 여러분들이 겪어보기를 바라는 취지로 냅니다. 예전 숙제 "어울리지 않아"의 정식 버전입니다. 여러분이 짜는 프로그램이 올바른 지를 확신할 수 있기를 바랍니다.

스트링은 0과 9사이의 정수들의 리스트이다: 예) 0000, 1102201, 998011199 등. 빈 스트링은 없다. "스트링 s가 코드 c와 어울린다"는 것은 코드 c가 표현하는 스트링 집합에 s가 포함된다는 뜻이다. 코드 c 는 다음과 같이 정의되고:

$$c \quad \rightarrow \quad 0 \mid 1 \mid \cdots \mid 9 \mid c \cdot c \mid c \mid c \mid c? \mid c*$$

코드 c가 뜻하는 스트링의 집합 $[\![c]\!]$는 다음과 같이 정의된다. 빈 스트링은 ϵ로 표현한다.

$$\begin{aligned}
[\![0]\!] &= \{0\} \\
&\vdots \\
[\![9]\!] &= \{9\} \\
[\![c_1 \cdot c_2]\!] &= \{s_1 s_2 \mid s_1 \in [\![c_1]\!], s_2 \in [\![c_2]\!]\} \\
[\![c_1 \mid c_2]\!] &= [\![c_1]\!] \cup [\![c_2]\!] \\
[\![c?]\!] &= \{\epsilon\} \cup [\![c]\!] \\
[\![c*]\!] &= \{\epsilon\} \cup [\![c]\!] \cup [\![c \cdot c]\!] \cup [\![c \cdot c \cdot c]\!] \cup \cdots
\end{aligned}$$

코드 데이타의 속내용을 감추는 다음의 함수들을 정의하고:

`atom` : int → code	`is-atom?` : code → bool
`dot` : code × code → code	`is-dot?` : code → bool
`bar` : code × code → code	`is-bar?` : code → bool
`optional` : code → code	`is-optional?` : code → bool
`star` : code → code	`is-star?` : code → bool
`de-atom` : code → int	`de-star` : code → code
`de-dot-0` : code → code	`de-bar-0` : code → code
`de-dot-1` : code → code	`de-bar-1` : code → code
`de-optional` : code → code	

하는 일은:

$$\begin{aligned}
\texttt{(de-atom (atom } n\texttt{))} &= n \\
\texttt{(de-dot-}i\texttt{ (dot } c_0\ c_1\texttt{))} &= c_i \\
\texttt{(de-bar-}i\texttt{ (bar } c_0\ c_1\texttt{))} &= c_i \\
\texttt{(de-star (star } c\texttt{))} &= c \\
\texttt{(de-optional (optional } c\texttt{))} &= c
\end{aligned}$$

위의 함수들을 이용해서, 주어진 스트링 s가 코드 c와 어울리는 지를 결정하는 함수

`match` : 스트링 × 코드 → bool

를 정의하라. 예를 들어, (match 11 1·0*·1)는 참을, (match 11 (10)*·1)은 거짓을 낸다. 스트링은 정수들의 리스트로 구현한다: 예) 1001은 '(1 0 0 1)로.
□