Homework 5
SNU 4910.210 Fall 2015

Chung-Kil Hur

due: 11/11(Wed) 23:59

이번 숙제의 목적은

- 데이터의 속 구현을 여러가지 방식으로 프로그램 해 보기.
- 속 구현이 여럿인 경우도 인터페이스만 알고 프로그램하는 것을 익히기.
- 올바른 프로그램인지 확신하기가 쉽지 않은 문제를 겪어보기. (여러분이 짜는 프로그램이 항상 올바른 답을 낸다는 것을 확신할 수 있기를 바랍니다.)
- 알아야 할 이론공부들이 많겠다는 동기를 가지게 하기.

Exercise 1 "안전한 합(sum) 구현"

다음의 만들기와 사용하기 함수들을 구현하여라. 그 구현은 type safe 해야 한다. 즉, 타입에 맞게 들어온 모든 입력에 대해 안전하게 타입에 맞는 값을 반환해야 한다.

- 만들기

$$\begin{aligned} \texttt{inl} &: \tau \to \tau + \tau' \\ \texttt{inr} &: \tau' \to \tau + \tau' \end{aligned}$$

- 사용하기

$$\texttt{case-sum} \;:\; (\tau \to \sigma) * (\tau' \to \sigma) * (\tau + \tau') \to \sigma$$

Exercise 2 "안전한 리스트(list) 구현"

다음의 만들기와 사용하기 함수들을 구현하여라. 그 구현은 type safe 해야 한다.

- 만들기

$$\begin{aligned} \texttt{empty} &: \tau\ list \\ \texttt{link} &: \tau * \tau\ list \to \tau\ list \end{aligned}$$

- 사용하기

$$\texttt{case-list} \ : \ (unit \to \sigma) * (\tau * \tau\ list \to \sigma) * \tau\ list \to \sigma$$

Exercise 3 "안전한 합과 리스트 사용"

앞에서 고안한 안전한 합과 리스트를 사용하여 아래의 함수들을 구현하고, 타입을 주석으로 달아서 안전한 프로그램인지 확인하라. 아래 함수들에서 합 타입은 오류 상황을 처리하기 위한 것이다. 함수에 (비록 타입에는 맞지만) 의도치 않은 입력이 들어왔을 경우, 합 타입의 오른쪽 값(즉, *unit* 타입의 값)을 반환하라.

$$\begin{aligned} \texttt{is-empty?} &: \tau\ list \to bool \\ \texttt{fst} &: \tau\ list \to \tau + unit \\ \texttt{rest} &: \tau\ list \to \tau\ list + unit \\ \texttt{length} &: \tau\ list \to int \\ \texttt{nth-elmt} &: \tau\ list * int \to \tau + unit \\ \texttt{map} &: (\tau \to \sigma) * \tau\ list \to \sigma\ list \\ \texttt{reduce} &: \tau\ list * (\tau * \sigma \to \sigma) * \sigma \to \sigma \end{aligned}$$

Exercise 4 "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 구현을 감추고 다음의 것들만 드러나도록 기획하였다. 각각을 구현하라.

$$\begin{aligned} &\texttt{black} : form \\ &\texttt{white} : form \\ &\texttt{glue} : form * form * form * form \to form \\ &\texttt{rotate} : form \to form \\ &\texttt{neighbor} : location * form \to int \\ &\texttt{pprint} : form \to void \end{aligned}$$

각각의 정의는 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 몇 가지가 있다:

- 방법 1. 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.

- 방법 2. 잎새에 기본 정사각형이 매달린, 모든 가지가 4갈래로 갈라지는 트리 구조로 표현하는 방법.

위 두 가지 구현 방안을 구현하라. 이 두 가지 구현 방안을 모두 가지고 위의 여섯가지(상수 두 개, 함수 네 개)를 정의하라. 이 때 데이타의 표현방식 두 가지가 적절히 모두 사용되도록 정의한다.

배열로 구현하는 경우, 드러나는(인터페이스) 함수들:

$$\begin{aligned}
\texttt{glue-array-from-tree}: &\quad form * form * form * form \to form \\
\texttt{glue-array-from-array}: &\quad form * form * form * form \to form \\
\texttt{rotate-array}: &\quad form \to form \\
\texttt{neighbor-array}: &\quad location * form \to int \\
\texttt{pprint-array}: &\quad form \to void \\
\texttt{is-array?}: &\quad form \to bool
\end{aligned}$$

트리로 구현하는 경우, 드러나는 함수들:

$$\begin{aligned}
\texttt{glue-tree-from-tree}: &\quad form * form * form * form \to form \\
\texttt{glue-tree-from-array}: &\quad form * form * form * form \to form \\
\texttt{rotate-tree}: &\quad form \to form \\
\texttt{neighbor-tree}: &\quad location * form \to int \\
\texttt{pprint-tree}: &\quad form \to void \\
\texttt{is-tree?}: &\quad form \to bool
\end{aligned}$$

두 구현사이의 변환 함수들:

$$\begin{aligned}
\texttt{array-to-tree}: &\quad form \to form \\
\texttt{tree-to-array}: &\quad form \to form
\end{aligned}$$

□

Exercise 5 "벽지 아가씨 심사위원"

벽지 무늬를 다루는 함수들에 다음의 함수를 추가로 정의하고:

$$\begin{aligned}
&\texttt{equal} : form * form \to bool \\
&\texttt{size} : form \to int
\end{aligned}$$

`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.

$$\texttt{beautiful} : \mathit{form} \to \mathit{bool}$$

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