Homework 5
SNU 4910.210 Fall 2012

Kwangkeun Yi

due: 10/26(Fri) 24:00

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 "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.

`black` : *무늬*
`white` : *무늬*
`glue` : *무늬* $*$ *무늬* $*$ *무늬* $*$ *무늬* $\rightarrow$ *무늬*
`rotate` : *무늬* $\rightarrow$ *무늬*
`neighbor` : *위치* $*$ *무늬* $\rightarrow$ *int*
`pprint` : *무늬* $\rightarrow$ *void*

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:

- 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.
- 등등

이 두 가지 구현 방안을 구현하라. 이 두 가지 구현 방안을 모두 가지고 위의 여섯가지 함수를 정의하라. 이 때 데이타의 표현방식 두 가지가 적절히 모두 사용되도록 정의한다.

배열로 구현하는 경우, 드러나는(인터페이스) 함수들:

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

트리로 구현하는 경우, 드러나는 함수들:

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

□

Exercise 2 "벽지 아가씨 심사위원"

벽지 무늬를 다루는 함수들에 다음의 함수를 추가로 정의하고:

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

`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.

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

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

Exercise 3 "튜링기계(Turing machine)"

튜링기계에 대한 아래글을 읽고 튜링기계를 구현해 봅시다.

"튜링기계(Turing Machine)의 고안"
`ropas.snu.ac.kr/~kwang/paper/cs-ch1.pdf`

아래의 함수들을 만들어서 제출합니다.

- 테잎을 만들고 사용하는 함수들:

  `init-tape` : *symbol list* → *tape*
  `read-tape` : *tape* → *symbol*
  `write-tape` : *tape* ∗ *symbol* → *tape*
  `move-tape-left` : *tape* → *tape*
  `move-tape-right` : *tape* → *tape*
  `print-tape` : *tape* → *void*

  테입에 기록되는 *symbol*은 문자열입니다. 읽기쓰기 헤드를 오른쪽/왼쪽으로 한 칸 움직이기 위해서 테입을 반대방향인 왼쪽/오른쪽으로 한 칸을 움직여 주는 함수들을 사용하게됩니다.

- 작동규칙표를 만들고 사용하는 함수들:

  `empty-ruletable` : *ruletable*
  `add-rule` : *rule* ∗ *ruletable* → *ruletable*
  `make-rule` : *state* ∗ *symbol* ∗ *todo* ∗ *move* ∗ *state* → *rule*
  `match-rule` : *state* ∗ *symbol* ∗ *ruletable* → *todo* × *move* × *state*

  튜링머신의 상태를 나타내는 *state*는 문자열(string)입니다. *todo*는 `'erase` 거나 `'write`와 *symbol*의 cons-쌍이고, *move*는 `'left`, `'right`, 혹은 `'stay`입니다.

- 튜링기계를 만들고 사용하는 함수들:

  `make-tm` : *symbol list* ∗ *state* ∗ *ruletable* → *tm*
  `step-tm` : *tm* → *tm*
  `run-tm` : *tm* → *tm*
  `print-tm` : *tm* → *void*

함수 `make-tm`는 주어진 심볼들을 테잎에 순서대로 써서 초기화하고, 테입의 첫번째 심볼의 위치에 읽기쓰기 헤드가 위치하고, 주어진 초기상태로 기계상태가 셋팅되고, 주어진 작동규칙표를 갖추게됩니다.

함수 `step-tm`은 튜링기계의 작동규칙표에 따라 주어진 튜링기계를 한 스텝 실행시키고, `run-tm`은 실행이 끝날 때까지 실행시키고 끝나는 경우 최종 튜링기계를 내놓습니다. □

Exercise 4 "어울리지 않아"

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

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

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

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

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

$$\texttt{smatch} : \text{스트링} * \text{코드} \rightarrow \mathit{bool}$$

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

Exercise 5 "즐거운 고민"

영희의 고민은 조카들이 모두 행복해 할 수 있도록 가장 저렴하게 선물을 준비하는 것입니다. 영희의 조카들은 시샘이 많습니다. 매년 이맘때쯤이면 영희는 조카들에게 선물을 한 꾸러미씩 나눠주는데, 받고나면 조카들끼리 다른 형제들이 받은 선물을 시샘하면서 서로 조르고 울고. 그래서 다시 정리해서 주면 또 만족스럽지 않아서 조르고 울고.

영희는 그 고민을 다음과 같이 풀기로 했습니다. 선물 쇼핑을 나가기전에 조카들에게 올해 받을 선물의 후보들을 알려주고 각자는 그중의 부분집합을 선물로 받을 것이라고 선언합니다. 그러곤 조카들에게 각자가 만족할(싸우지않을) 조건을 얘기하라고 합니다. 영희는 가장 적은 비용으로 이러한 조건들을 모두 만족시키도록 선물꾸러미들을 준비합니다.

조카들의 조건들은 이런식입니다: "나는 최소한 만년필과 동생 영희가 받은 선물만큼은 받아야 해요." "나는 최소한 철수오빠와 숙희언니의 선물들에 공통된 것들 하고, 영숙이 선물중에서 CD빼 것은 가져야 해요" 등등. 예를 들어 A, B, C 세명의 조카가 있다면, 조건에 따라 받는 선물은 다음과 같지요:

- 샘만 많은 조카들은 아무것도 못받습니다. A: "최소한 B 만큼", B: "최소한 A 만큼", C: "최소한 B 만큼."
- 까다로운 조카들도 아무것도 못받습니다. A: "최소한 B 만큼에서 만년필 말고", B: "최소한 A 만큼에서 CD 말고", C: "최소한 B 만큼에서 USB 말고."
- 탐욕스런 조카들도 아무것도 못받습니다. A: "최소한 B와 C만큼", B: "최소한 A와 C만큼", C: "최소한 A와 B만큼."

- 샘이 없는 조카들은 원하는 것만 받습니다. A: "최소한 만년필", B: "최소한 CD", C: "최소한 USB."

조카의 조건은 다음과 같은 꼴로 표현된다고 정합시다:

"나는 최소한 ($cond_1$ 그리고 $\cdots$ 그리고 $cond_k$)을 받아야 해요."

이제 조카들의 조건들을 받아서 최소의 선물쇼핑 리스트를 작성하는 `shoppingList`를 작성하기 바랍니다:

$$\texttt{shoppingList} : (id \times cond)\ list \to (id \times gift\ \ list)\ list$$

결과는 조카마다 사주어야 할 선물들의 리스트입니다. 예를들어, 조카들의 조건이 다음과 같을때

$A:$ 최소한 ($\{1,2\}$ 하고 common(B, C))를 받아야.

$B:$ 최소한 common(C, $\{2,3\}$)를 받아야.

$C:$ 최소한 ($\{1\}$ 하고 (A except $\{3\}$))를 받아야.

최소의 선물꾸러미들은 A에게 $\{1,2\}$, B에게 $\{2\}$, C에게 $\{1,2\}$이므로, `shoppingList`의 결과는

```
((A . (1 2)) (B . (2)) (C . (1 2)))
```

입니다. 조카마다 받는 선물꾸러미는 "집합"입니다, 즉, 한 선물 꾸러미에는 같은 선물이 두개이상 포함되지는 않습니다.

구현을 위해서, 선물의 조건을 만드는 함수와 사용하는 함수는 다음과 같이 주어집니다.

만들기 함수들:

`mustItems` : $gift\ list \to cond$	가져야할 선물들
`mustBeTheSame` : $id \to cond$	어느 조카와 같아야하는지
`mustHaveExceptFor` : $cond * gift\ list \to cond$	조건에서 어느 선물들은 빼고
`mustHaveCommon` : $cond * cond \to cond$	두 조건에 공통된 선물들
`mustAnd` : $cond * cond \to cond$	두 조건 모두 만족해야

사용하기 함수들:

`isItems` : $cond \to bool$	`isSame` : $cond \to bool$
`isExcept` : $cond \to bool$	`isCommon` : $cond \to bool$
`isAnd` : $cond \to bool$	`whichItems` : $cond \to gift\ list$
`whoTheSame` : $cond \to id$	`condExcept` : $cond \to cond$
`itemsExcept` : $cond \to gift\ list$	`condCommon` : $cond \to cond \times cond$
`condAnd` : $cond \to cond \times cond$	

위에서 *gift*는 정수로, 조카 이름 *id*는 Scheme의 심볼로 구현됩니다. 예를 들어, 위에서 조카 A의 조건(*cond*)은 다음과 같이 만들어 지겠지요:

```
(mustAnd (mustItems '(1 2))
         (mustHaveCommon (mustBeTheSame 'B)
                         (mustBeTheSame 'C)))
```

□