

Homework 3

SNU 4910.210 Fall 2013

Kwangkeun Yi

due: 10/03(Thu) 24:00

이번 숙제의 목적은:

- 프로그램에서 정의되는 이름의 타입을 코멘트하는 것을 익힌다.
- 타입을 따지면서 프로그램하는 것을 익힌다.
- 데이터의 속 구현을 보지말고 인터페이스만 알고 프로그램하는 것을 익힌다.

Exercise 1 “타입 주석”

지난 숙제 1과 2의 문제중 1.1, 1.4, 1.5, 2.1, 2.2, 2.3 의 모범답안을 보고, `define`, `let`, `letrec`으로 정의한 이름들의 타입을 주석으로 달아서 제출한다. 주석을 다는 규칙은:

- 이름이 선언되는 그 줄에 그 이름의 타입을 주석으로 단다.
- 함수정의인 경우, 함수의 타입과 그 인자들의 타입을 주석으로 단다.
- 타입 주석은 강의노트에서 정한 타입언어를 사용한다.
- 주석은 한 줄로 제한한다.

예를들어:

```
(define (foo n) // foo: int -> int list, n: int
  (if (= n 0) ()
    (let ((rest (foo (- n 1)))) // rest: int list
      (cons n rest))))
```

□

Exercise 2 “미로 검증”

잡지에 가끔 미로 퀴즈가 부록으로 있었다. 종이에 그려진 미로를 상상해 보자. 그 미로를 다음과 같이 바라보자:

- 종이에 정사각형들이 빼곡히 채워져 있다(모눈종이). 각 정사각형은 하나의 방이다.
- 각 방들은 이웃한 방들과 사이의 벽들이 몇 개 터져 있기도 하고 막혀있기도 하다.
- 시작 방과 끝 방이 정해져 있다.

미로퀴즈를 잡지에 출판하기에 앞서, 편집진은 과연 미로퀴즈의 답이 있는 지 확인하는 과정을 밟을것이다. 시작 방에서 끝 방으로 이어지는 길이 있는지.

그러한 검증을 하는 `maze-check` 함수를 정의해 보자. (이러한 검증함수는 실제로 미로를 찾아내 주는 함수보다 간단하다):

$$\text{maze-check} : \text{maze} \times \text{room} \times \text{room} \rightarrow \text{bool}$$

`maze-check`은 미로와 시작 방과 끝 방을 주면 그 두 방을 연결하는 길이 있는 지를 확인해 준다. 이때 미로는 유한하고 시작 방과 끝 방은 항상 그 미로안에 있는 방이라고 가정한다.

위의 함수를 구현 할 때는 미로가 어떻게 구현되었는지, 집합은 어떻게 구현되었는 지 알지 못하는 상태에서 다음을 사용해서 구현할 수 있다:

$$\text{can-enter} : \text{room} \times \text{maze} \rightarrow \text{room list}$$
$$\text{same-room?} : \text{room} \times \text{room} \rightarrow \text{bool}$$
$$\text{empty-set} : \text{room set}$$
$$\text{add-element} : \text{room} \times \text{room set} \rightarrow \text{room set}$$
$$\text{is-member?} : \text{room} \times \text{room set} \rightarrow \text{bool}$$
$$\text{is-subset?} : \text{room set} \times \text{room set} \rightarrow \text{bool}$$

`can-enter`는 미로의 주어진 방에서 갈 수 있는 이웃한 방들의 리스트를 준다. `same-room?`은 두 방이 같은 방인지를 판별해 준다. 위의 여섯 함수들은 이번 숙제에서는 구현하지 않는다. □.

Exercise 3 “미로 만들기”

$n \times m$ (x 축, y 축)개의 정육각형 방을 가진 모눈종이를 생각하자. 육각형 방은 항상 가로로 누운 육각형이고, 각 미로의 행은 제일 왼쪽 육각형방의 동북쪽 벽이 다음 육각형방으로 연결되면서 시작한다(아래 육각미로의 예 참고).

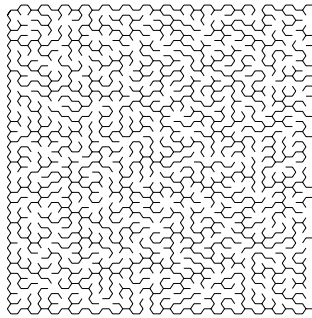
미로의 정의는: 입구와 출구가 맨 윗줄의 방과 맨 아랫줄의 방에 각각 하나
이고, 입구에서 출구까지의 통로가 존재하는 것을 뜻한다.

미로를 만드는 함수

`mazeGen : int × int → maze`

를 작성하라. 즉, `(mazeGen n m)` 은 양수 n (x축, 열의 수)과 m (y축, 행의 수)
을 받아 $n \times m$ 육각모눈종이에 미로를 만들어 준다.

육각미로의 한 예:



다음의 함수들을 사용해서 정의하라. 이 함수들은 이번숙제에서는 정의하
지 않는다.

`init-maze : int × int → maze`
`open-s : int × int × maze → maze`
`open-n : int × int × maze → maze`
`open-se : int × int × maze → maze`
`open-sw : int × int × maze → maze`
`open-ne : int × int × maze → maze`
`open-nw : int × int × maze → maze`
`maze-pp : maze → 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 된
다. `(open-d n m M)`는 미로 M 에서 방 (n, m) 의 d -방향 벽을 뜯은 미로를 만
든다. `(maze-pp M)`는 미로 M 을 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.

채점기준: 미로를 만들기만 하면 만점. □