

Homework 2

SNU 4910.210 Fall 2012

Kwangkeun Yi

due: 9/21(Fri) 24:00

이번 숙제의 목적은:

- 재귀적인 프로그램을 익숙하게 한다.
- 재귀적인 데이터 생성 함수들을 이용해 프로그램하는 것을 익힌다.
- 타입을 따지면서 프로그램하는 것을 익힌다.

Exercise 1 “나무구조 데이터”

컴퓨터과학(computer science)은 나무를 많이 다룬다. 깊고 검은 숲이나, 아름드리 나무나, 왕성하게 뻗은 가지들을 항상 다루게 된다.

나무구조(혹은 가지구조, tree)는 다음과 같이 정의된다:

- 기본 나무구조: 잎새 하나는 나무구조이다.
- 나무구조들을 품고있는 나무구조: 하나의 꼭지에서 한 개 이상의 나무구조들이 하나하나 가지로 뻗어나간 구조는 다시 나무구조이다.

위의 두 가지 조건이 나무구조를 만드는 두가지 방법을 결정한다: 기본적인 나무를 만드는 방법(base case)과 만든 나무들을 가지고 새로운 나무를 만드는 방법(inductive case).

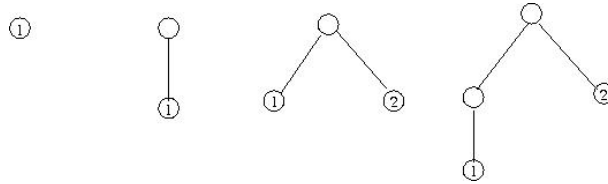
나무구조를 만드는 다음의 두 함수를 정의하라:

$$\begin{aligned} \text{leaf} &: \tau \rightarrow \text{tree} \\ \text{node} &: \text{tree list} \rightarrow \text{tree} \end{aligned}$$

`leaf`는 어떤 타입(τ)의 값을 가지고 그 값의 잎새 나무를 만든다. 즉, (`leaf 1`)하면 정수 1을 가지는 잎새 나무가, (`leaf 'a`)하면 심볼 `a`를 가지는 잎새 나무가, (`leaf '(1 2)`)하면 리스트 `(1 2)`를 가지는 잎새나무가 되겠다. `node`는

나무들의 리스트를 받아서 그들을 차례로 매달고 있는 새로운 나무를 만든다.
 node는 빈 리스트를 받으면 정의되지 않는다.

예를 들어, 아래 그림의 나무구조들은



각각 (leaf 1), (node (list (leaf 1))), (node (list (leaf 1) (leaf 2))),
 그리고 (node (list (node (list (leaf 1))) (leaf 2))) 로 만든 구조물
 들이 되겠다.

나무를 사용하는 다음 세가지 함수들도 정의하라:

```
is-leaf? : tree → bool
leaf-val : tree → τ
nth-child : tree × nat → tree
```

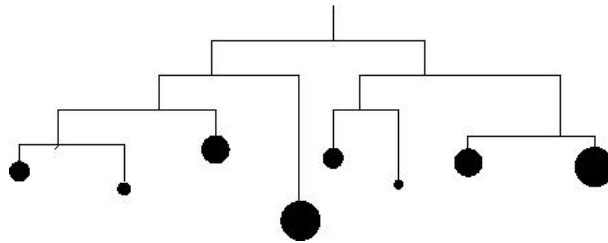
is-leaf?는 주어진 나무가 잎새 나무인지 아닌지를 판별한다. leaf-val은 잎
 새가 가지고 있는 데이터를 꺼낸다. 그 데이터의 타입이 τ . nth-child는 나무
 와 자연수 $n \geq 0$ 을 받아서 그 나무의 n 번째 가지의 나무(n -th subtree)를 내 놓
 는다. (N 개는 0번째부터 $N - 1$ 번째가 있다.) □

Exercise 2 “모빌 무게 재기”

천장에 매달려 균형을 잡은 채 은은히 흔들리고 있는 모빌을 떠올려보자.
 일반적인 두갈래 모빌은 다음과 같이 정의된다:

- 모형 하나는 모빌구조이다.
- 모빌들을 품고있는 모빌: 하나의 균형점에서 왼쪽/오른쪽의 모빌구조들
 이 뺀어나간 구조는 다시 모빌구조이다.

예를 들어 두 갈래 모빌의 한 예는 아래와 같다:



모빌구조를 만드는 다음 세 가지 함수를 정의하라. 단, 반드시 Exercise 1에서 정의한 함수를 사용한다:

```

model : nat → mobil
make-branch : nat × mobil → branch
make-mobil : branch × branch → mobil

```

model은 입력한 자연수값만큼의 무게를 갖는 모형 모빌을 만든다. make-branch는 그 길이와 끝에 매달린 모빌을 받아 하나의 가지를 이룬다. make-mobil은 왼쪽/오른쪽의 가지를 받아 하나의 모빌을 이룬다.

모빌을 사용하는 다음 두 가지 함수들도 정의하라:

```

is-balanced? : mobil → bool
weight : mobil → nat

```

is-balanced?는 주어진 모빌이 “균형잡혀있는지”를 판별한다. 하나의 모빌은 양쪽 가지의 토크(길이×무게)가 같을 때 균형이 잡히며, 두 가지의 무게의 합이 그 모빌의 무게가 된다. 한 모빌 구조 내의 모든 하위 모빌 구조가 균형 상태에 있을 때, 그 모빌은 “균형잡혀있다”고 한다. weight는 그 모빌구조의 총 무게를 내놓는다. □

Exercise 3 “논리회로”

부울 회로(Boolean circuit)라는 것은 다음과 같이 귀납적으로 정의된다. 모든 부울 회로는 하나 이상의 입력과 하나의 출력을 가지고 있다.

- 기본: 0을 가진 전기줄은 부울 회로이다.
- 기본: 1을 가진 전기줄은 부울 회로이다.
- 귀납: 부울 회로 하나를 가지고 또다른 회로를 만드는 방법 not이 있다.
- 귀납: 부울 회로 두개를 가지고 또다른 회로를 만드는 방법 and가 있다.
- 귀납: 부울 회로 두개를 가지고 또다른 회로를 만드는 방법 or가 있다.

부울 회로를 만드는 위의 다섯가지 방법들을 정의하라:

```

zero : circuit
one : circuit
not-circuit : circuit → circuit
and-circuit : circuit × circuit → circuit
or-circuit : circuit × circuit → circuit

```

위의 것들을 정의할 때는 반드시 Exercise 1에서 정의한 함수만을 사용한다.

그리고 부울 회로를 사용하는 아래의 여섯가지 함수들도 정의하라:

```
is-zero? : circuit → bool
is-one?  : circuit → bool
is-not?  : circuit → bool
is-and?  : circuit → bool
is-or?   : circuit → bool
sub-circuit : circuit × nat → circuit
```

`sub-circuit`은 회로와 자연수 $n \geq 0$ 을 받아서 그 회로의 n 번째 가지의 회로(n -th sub-circuit)를 내 놓는다. □

Exercise 4 “논리 회로의 계산”

Exercise 3의 함수들로 만들어진 부울 회로의 최종 출력값(회로의 의미)를 계산하는 함수

```
output : circuit → {0,1}
```

를 정의하라.

부울 회로의 최종 출력값은 그 회로가 어떻게 만들어 졌냐에 따라 다음과 같이 재귀적으로 정의된다. `zero`의 출력값은 0. `one`의 출력값은 1. `(not B)`은 회로 B 의 출력값이 0이면 1, 1이면 0. `(and B1 B2)`은 회로 B_1 과 B_2 둘의 출력값이 모두 1일 때만 1, 아니면 0. `(or B1 B2)`은 회로 B_1 과 B_2 둘의 출력값이 모두 0일 때만 0, 아니면 1. □