Homework 6
SNU 4910.210 Fall 2012

Kwangkeun Yi

due: 11/20(Tue) 24:00

이번 숙제의 목적은

- 타입으로 프로그램을 정리하는(typeful programming) 연습하기.
- 타입으로 프로그램을 정리하는 "즐거운" 환경에서 프로그램 연습하기.
  즉, 자동으로 타입 검증을 해주는 환경에서 프로그래밍 연습하기.

Exercise 1 "대진표 스트링"

일반적으로 게임 대진표는 완전한 이진 나무구조(complete binary tree)입니다. 2014 얼드컵 팀들과 그 대진표를 다음과 같이 정의했습니다:

```
type team = Korea | France | Usa | Brazil | Japan | Nigeria | Cameroon
          | Poland | Portugal | Italy | Germany | Norway | Sweden | England
          | Argentina
type tourna = LEAF of team
            | NODE of tourna * tourna
```

`tourna`를 받아서 괄호를 이용한 1차원 스트링으로 변환해주는 함수 `parenize`를 작성하세요:

```
parenize: tourna -> string
```

예를들어,

```
parenize(NODE(NODE(LEAF Korea, LEAF Portugal), LEAF Brazil))
= "((Korea Portugal) Brazil)"
```

□

Exercise 2 "탈락"

이제 다음의 함수, drop을 작성하라:

```
drop: tourna * team -> string
```

drop(t, Brazil)는 축구대진표에서 Brazil 팀이 탈락한 경우 새롭게 구성되는 대진표를 출력한다(위의 toParen을 사용). □

Exercise 3 "참거짓"

Propositional Logic 식들(formula)을 다음과 같이 정의했다:

```
type formula = TRUE
             | FALSE
             | NOT of formula
             | ANDALSO of formula * formula
             | ORELSE of formula * formula
             | IMPLY of formula * formula
             | LESS of expr * expr
and  expr = NUM of int
          | PLUS of expr * expr
          | MINUS of expr * expr
```

주어진 formula를 받아서 참값을 만들어내는 함수 eval

```
eval: formula -> bool
```

를 정의하라. □

Exercise 4 "자연수"

자연수 nat 는 다음과 같이 정의될 수 있다:

```
type nat = ZERO | SUCC of nat
```

두 자연수를 받아서 그 합/곱에 해당하는 자연수를 만드는 두 함수

```
natadd : nat * nat -> nat
natmul : nat * nat -> nat
```

를 정의하세요. □

Exercise 5 (10pts) "Mathemadiga"

고등학교때는 손으로하고, Maple이나 Mathematica에서는 자동으로 해주던 미분식 전개를 만들어봅시다.

```
diff: ae * string -> ae
```

`diff`는 식(algebraic expression)과 변수를 받아서 주어진 식을 변수로 미분한 결과 식을 돌려 줍니다. 예를들어, 식 $ax^2 + bx + c$을 x에 대해 미분시키면 $2ax + b$를 내 놓는 것입니다. 미분된것을 될 수 있으면 최소의 꼴로 줄이거나 등등의 작업을 하는 것은 자유입니다. 미분할 식은 다음의 `ae` 타입입니다:

```
type ae = CONST of int
        | VAR of string
        | POWER of string * int
        | TIMES of ae list
        | SUM of ae list
```

□

Exercise 6 "k-친수"

일반적으로 k진수($k > 1$)는 다음과 같이 표현한다.

$$d_0 \cdots d_n$$

여기서

$$\forall d_i \in \{0, \cdots, k-1\}.$$

그리고 "$d_0 \cdots d_n$"은 크기가

$$d_0 \times k^0 + \cdots + d_n \times k^n$$

인 정수를 표현한다.

이것을 살짝 확장해서 "k친수"를 다음과 같이 정의해보자. 표현은

$$d_0 \cdots d_n$$

여기서

$$\forall d_i \in \{1-k, \cdots, 0\} \cup \{0, \cdots, k-1\}.$$

그리고 "$d_0 \cdots d_n$"은 크기가

$$d_0 \times k^0 + \cdots + d_n \times k^n$$

인 정수를 표현한다.

예를 들어, 2친수의 경우를 생각하자. 베이스가 $\{-1, 0, 1\}$이 되겠다. 0이 0을, +가 1을 -가 -1을 표현한다고 하면, + 는 1을, +0+는 5를, +-는 -1을, +-0-는 -9인 정수를 표현한다.

OCaml로 2친수라는 타입을 다음과 같이 정의했다:

```
type crazy2 = NIL | ZERO of crazy2 | ONE of crazy2 | MONE of crazy2
```

예를 들어, 0+-은

```
ZERO(ONE(MONE NIL))
```

로 표현된다.

위와 같이 표현되는 2친수를 받아서 그것의 값을 계산하는 함수 crazy2val을 정의하세요.

```
crazy2val: crazy2 -> int.
```

□

Exercise 7 "2친수 더하기"

두 2친수를 받아서 2친수의 합에 해당하는 2친수를 내어놓는 함수 crazy2add를 정의하세요.

```
crazy2add: crazy2 * crazy2 -> crazy2
```

위의 crazy2add는 다음의 성질이 만족되야 한다: 임의의 2친수 z 과 z'에 대해서

$$\texttt{crazy2val (crazy2add}(z,z')) = \texttt{crazy2val}(z) + \texttt{crazy2val}(z').$$

□

Exercise 8 "CheckMetroMap"

아래 metro 타입을 생각하자:

```
type metro = STATION of name
           | AREA of name * metro
           | CONNECT of metro * metro
and name = string
```

아래 checkMetro 함수를 정의하라:

```
checkMetro: metro -> bool
```

checkMetro는 주어진 metro 가 제대로 생겼는지를 확인해 준다. "metro가 제대로 생겼다"는 것은(iff) 메트로 역 이름(id in STATION(id))들이 항상 자기 이름의 지역(m in AREA(id, m))에서만 나타나는 경우를 뜻한다.

예를들어, 제대로 생긴 metro 들은:

- AREA("a", STATION "a")
- AREA("a", AREA("a", STATION "a"))
- AREA("a", AREA("b", CONNECT(STATION "a", STATION "b")))
- AREA("a", CONNECT(STATION "a", AREA("b", STATION "a")))

그렇지 못한 것들의 예들은:

- AREA("a", STATION "b")
- AREA("a", CONNECT(STATION "a", AREA("b", STATION "c")))
- AREA("a", AREA("b", CONNECT(STATION "a", STATION "c")))

□