Homework 6
SNU 4910.210 Fall 2014

Chung-Kil Hur

due: 11/23(Sun) 24:00

이번 숙제의 목적은

- 명령형 방식(imperative programming, stateful programming) 연습하기.
- 타입으로 프로그램을 정리하는(typeful programming) 연습하기.
- 타입으로 프로그램을 정리하는 "즐거운" 환경에서 프로그램 연습하기. 즉, 자동으로 타입 검증을 해주는 환경에서 프로그래밍 연습하기.
- 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 "보급로 갯수"

보급로 갯수를 세는 문제를 떠올려보자. $N \times N$개의 칸을 가지는 바둑판 좌표에서, 시작점 $(0,0)$에서 (n,m)까지의 최단거리 길의 갯수를 세는 함수 `ways`를 재귀적으로 정의하면 다음과 같다:

```
(define (ways n m)
  (cond ((= n 0) 1)
        ((= m 0) 1)
        (else (+ (ways (- n 1) m)
                 (ways n (- m 1))))))
```

그러나 이 재귀함수는 `(ways 50 50)`의 실행에도 쉽사리 답을 내놓지 않을만큼 효율이 낮다. 교재 3.3.3장과 연습문제 3.27을 참조하여 재귀과정에서

먼저 계산한 값을 기억해두었다가 재사용하는 함수 memo-ways를 만들어보자:

$$\texttt{memo-ways} : nat \times nat \to nat$$

□

Exercise 2 "대진표 스트링"

일반적으로 게임 대진표는 완전한 이진 나무구조(complete binary tree)입니다. 2013 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
```

tourna를 받아서 괄호를 이용한 1차원 스트링으로 변환해주는 함수 parenize를 작성하세요:

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

예를들어,

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

□

Exercise 3 "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 4 "참거짓"

선언논리(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 5 (10pts) "Mathemadiga"

고등학교때는 손으로하고, Maple이나 Mathematica에서는 자동으로 해주던 미분식 전개를 만들어봅시다.

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

diff는 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. `0`이 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 `crazy2val`을 정의하세요.

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

□

Exercise 7 "2친수 더하기"

두 2친수를 받아서 2친수의 합에 해당하는 2친수를 내어놓는 함수 `crazy2add`를 정의하세요.

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

위의 `crazy2add`는 다음의 성질이 만족되야 한다: 임의의 2친수 z 과 z'에 대해서

`crazy2val (crazy2add(`z`,`z'`))` $=$ `crazy2val(`z`)` $+$ `crazy2val(`z'`)`.

□

Challenge 2 "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 CD뺀 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` : *giftlist* → *cond*	가져야할 선물들
`mustBeTheSame` : *id* → *cond*	어느 조카와 같아야하는지
`mustHaveExceptFor` : *cond* ∗ *giftlist* → *cond*	조건에서 어느 선물들은 빼고
`mustHaveCommon` : *cond* ∗ *cond* → *cond*	두 조건에 공통된 선물들
`mustAnd` : *cond* ∗ *cond* → *cond*	두 조건 모두 만족해야

- 사용하기 함수들:

`isItems` : *cond* → *bool*	`isSame` : *cond* → *bool*
`isExcept` : *cond* → *bool*	`isCommon` : *cond* → *bool*
`isAnd` : *cond* → *bool*	`whichItems` : *cond* → *giftlist*
`whoTheSame` : *cond* → *id*	`condExcept` : *cond* → *cond*
`itemsExcept` : *cond* → *giftlist*	`condCommon` : *cond* → *cond* × *cond*
`condAnd` : *cond* → *cond* × *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)))` □