Homework 6
SNU 4910.210 Fall 2015

Chung-Kil Hur

due: 11/22(Sun) 23:59

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 "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")))

□

Exercise 7 "미라쥐 언어"

다음과 같은 언어 Mirage 를 생각합시다.

program	→	c	
c	→	x := e	assignment
	|	c ; c	sequence
	|	c^*	non-deterministic 0 or more repetitions
	|	$c \cup c$	non-deterministic choice
	|	x eq e ? c	check
	|	x neq e ? c	check
e	→	n	integer
	|	$e + e$	addition
	|	$e - e$	substraction
	|	x	

각 명령어는 기계상태를 변환시켜줍니다. 기계는 프로그램 변수의 정수값을 가지고 있습니다. 프로그램 변수는 오직 하나만 있습니다.

- "$x := e$"는 현재 기계상태에서 e를 계산해서 x의 값으로 기계상태를 변환
- "c_1 ; c_2"는 c_1 실행 후 결과 상태를 가지고 c_2를 실행
- "c^*"는 c를 0번 이상 임의의 횟수를 반복
- "$c_1 \cup c_2$"는 c_1이나 c_2중 하나를 선택해서 실행
- "x `eq` e ? c"는 현재상태에서 x의 값이 e와 같은 지 확인. 같으면 c를 실행, 틀리면 건너뜀
- "x `neq` e ? c"는 현재상태에서 x의 값이 e와 같은 지 확인. 다르면 c를 실행, 같으면 건너뜀
- 변수가 가질 수 있는 값은 -5 에서 +5 까지의 정수. 그 이외의 값을 변수가 가지게 되면 현재상태로 실행 끝

Mirage 프로그램의 싹쓸이(exhaustive) 실행기 `exeval`

$$\texttt{exeval} : \texttt{pgm} -> \texttt{state} -> \texttt{statelist}$$

을 작성합시다. "싹쓸이"란 모든 가능한 경우를 실행하는 것을 말합니다.

```
type pgm = cmd
and cmd = ASSIGN of exp
        | SEQUENCE of cmd * cmd
        | REPEAT of cmd
        | CHOICE of cmd * cmd
        | EQ of exp * cmd
        | NEQ of exp * cmd
and exp = NUM of int
        | ADD of exp * exp
        | SUB of exp * exp
        | VAR
type state = int
```

예를들어,

```
x := 1; ((x eq 1? x := x+1) U (x neq 1? x := x-1))*
```

를 실행하면 결과 상태들은 [1, 2].

또다른 예로,

```
x := 1; (x := x+1)*
```

를 실행하면 결과 상태는 [1,2,3,4,5].

□