Homework 1
SNU 4190.310, Fall 2010
Kwangkeun Yi
Due: 9/15(Wed), 24:00

Exercise 1 "씨그마"

우리가 중고등 수학시간에 숱하게 썼던 다음의 "씨그마"를 OCaml로 정의하세요:

$$\Sigma_{n=a}^{b} f(n)$$

씨그마의 타입은

```
sigma : int * int * (int -> int) -> int.
```

즉, `sigma(a,b,f)`로 표현하면 $\Sigma_{n=a}^{b} f(n)$과 같도록. □

Exercise 2 "합곱"

아래의 성질을 만족하는 함수 `sumprod`를 정의하세요:

$$\texttt{sumprod}(matrix, \texttt{n}, \texttt{k}) = \Sigma_{i=1}^{n} \Pi_{j=1}^{k} matrix(i,j).$$

즉, `sumprod`의 타입은

```
sumprod : (int * int -> real) * int * int -> real
```

□

Exercise 3 "말해줘"

114 전화 시스템에서는 찾은 전화번호를 음성합성으로 알려주는데, 이 시스템의 핵심함수, `vocalize`를 작성하세요:

```
vocalize: string -> string list list
```

일곱 혹은 여덟 자 양의 정수 string을 받아서 세자리와 네자리(혹은 네자리와 네자리)로 나누어서 읽어주도록 하는 함수입니다. 예를 들어,

```
vocalize "8801857"
= [["팔","백","팔","십"],["천","팔","백","오","십","칠"]]
```

단, 0을 읽지않아도 될 경우 읽지않습니다. 예를들어, 0022900은

```
[["이"], ["이", "천", "구", "백"]]
```

□

Exercise 4 "대진표 스트링"

일반적으로 게임 대진표는 완전한 이진 나무구조(complete binary tree)입니다. 2006 얼드컵 팀들과 그 대진표를 다음과 같이 정의했습니다:

```
type team = Korea | France | Usa | Brazil | Japan | Nigeria | Cameroon
          | Poland | Portugal | Italy | Germany | Sweden | England
          | Croatia | 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 5 "참거짓"

Propositional Logic 식들(`formula`)을 다음과 같이 정의했습니다:

```
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

$$\texttt{eval} : \texttt{formula} \rightarrow \texttt{bool}$$

를 정의하세요. □

Exercise 6 "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 7 "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 8 "짚-짚-나무"

임의의 나무를 여러분 바지의 "지퍼"로 구현할 수 있답니다.

- 나무구조 타입은 아래와 같이 정의되겠지요:

```
type tree = LEAF of item
          | NODE of tree list
```

- 아래의 `zipper`가 나무의 줄기를 타고 자유자재로 찢어놓기도 하고 붙여놓기도 합니다.

```
type zipper = TOP
            | HAND of tree list * zipper * tree list
```

현재 나무줄기의 어느지점에 멈춰 있는 지퍼손잡이 `HAND(l,z,r)`에서, `l`은 왼편 형제 나무들(elder siblings)이고 `r`은 오른편 형제 나무들(younger siblings)이다.

- 나뭇가지에서의 현재 위치 `location`는 현재위치를 뿌리로하는 나무자체와 지퍼(`zipper`)로 표현되는 주변 나무들로 구성된다.

```
type location = LOC of tree * zipper
```

- 예를들어, "$a \times b + c \times d$" 가 다음과 같은 나무구조로 표현될 것이다. 모든 심볼은 항상 잎새에 매달리게 된다.

```
NODE [ NODE [LEAF a; LEAF *; LEAF b];
       LEAF +;
```

```
                NODE [LEAF c; LEAF *; LEAF d]
              ]
```

두번째 곱셈표에의 위치는 다음과 같다:

```
LOC (LEAF *,
     HAND([LEAF c],
          HAND([LEAF +; NODE [LEAF a; LEAF *; LEAF b]],
               TOP,
               []),
          [LEAF d]))
```

- 자, 주어진 위치에서 이제 자유자재로 나무를 탈 수 있습니다. 왼편으로 옮겨가는 것은 다음과 같지요:

```
let goLeft loc = match loc with
    LOC(t, TOP) -> raise NOMOVE "left of top"
  | LOC(t, HAND(l::left, up, right)) -> LOC(l, HAND(left, up, t::right))
  | LOC(t, HAND([],up,right)) -> raise NOMOVE "left of first"
```

- 다음의 나머지 함수들을 정의하세요:

```
goRight: location -> location
goUp: location -> location
goDown: location -> location
```

□