

Homework 3
SNU 4190.310, 2006 가을
이 광근
due: 10/13(Fri), 24:00

Exercise 1 (30pts) “K- 실행기 II”

타입 선언을 제외한 K- 실행기를 완성하자. 지난 숙제의 K-에 다음이 첨가된 경우이다: 함수(procedure) + 레코드(record) + 포인터(pointer).

아래의 KMINUS 꼴을 가지는 모듈 K를 정의한다.

```
signature KMINUS =  
sig  
  exception Error of string  
  type id = string  
  type exp = NUM of int | TRUE | FALSE | UNIT  
            | VAR of id  
            | ADD of exp * exp  
            | SUB of exp * exp  
            | MUL of exp * exp  
            | DIV of exp * exp  
            | EQUAL of exp * exp  
            | LESS of exp * exp  
            | NOT of exp  
            | ASSIGNV of id * exp          (* assign to variable *)  
            | ASSIGNF of exp * id * exp   (* assign to record field *)  
            | ASSINGG of exp * exp        (* generic assign *)  
            | SEQ of exp * exp            (* sequence *)
```

```

| IF2 of exp * exp * exp      (* if-then-else *)
| IF1 of exp * exp           (* if-then *)
| WHILE of exp * exp         (* while loop *)
| FOR of id * exp * exp * exp (* for loop *)
| LETV of id * exp * exp      (* variable binding *)
| LETF of id * id * exp * exp (* procedure binding *)
| CALLV of id * exp          (* call by value *)
| CALLR of id * id           (* call by reference *)
| RECORD of (id * exp) list  (* record construction *)
| FIELD of exp * id          (* record field selection *)
| MALLOC of exp              (* malloc *)
| AMPER of id                 (* ampersand x *)
| STAR of exp                 (* *E *)
| READ of id
| WRITE of exp

type program = exp
type memory
type env
type value
val emptyMemory: memory
val emptyEnv: env
val run: memory * env * program -> value
end

```

K- 프로그램이 어떻게 exp들로 표현될지는 쉽게 추측할 수 있을 것입니다. exp으로 표현된 K- 프로그램이 S 라고 하면,

$$K.run (K.emptyMemory, K.emptyEnv, S)$$

는 프로그램 S 를 실행시키게 되는데, 성공적으로 끝나면 최후의 값을 내어주게 됩니다. 이때 프로그램은 실행중에 I/O를 하면서 프로그램이 하는 일을 바깥세상에 드러내게 되겠지요. 실행중에 타입이 맞지 않는 프로그램이면 Error라는 예외상황을 발생시키고 프로그램 실행이 중단되어야 합니다. “Error”란 (if and only if) 강의시간에 정의된 의미 규칙으로는 그 프로그램의 의미가 정의될 수 없는 경우입니다. 입출력은 정수만 가능합니다. 출력은 정수를 화면에 뿌리고 “newline”을 프린트합니다. □

Exercise 2 (30pts) “K- 프로그래밍”

다음을 K- 로 작성하고, 위에서 구현한 실행기 K.run로 실행시켜 제대로 실행되는 지를 확인한다.

큐를 두개의 스택으로 구현하는 것을 작성합니다. 큐는 반드시 하나의 리스트일 필요는 없어요. 두개의 스택을 이용해서 큐에 넣고 빼는 작업이 거의 한 스텝만으로 실현될 수 있습니다. 각각의 큐 연산들의 타입들은:

```
emptyQ: Q
enQ: Q * int -> Q
deQ: Q -> int * Q
```

큐를 $[a_1, \dots, a_m, b_1, \dots, b_n]$ 라고 합시다 (b_n 이 머리). 이 큐를 두개의 리스트 L 과 R 로 표현할 수 있습니다:

$$L = [a_1, \dots, a_m], \quad R = [b_n, \dots, b_1].$$

한 원소 x 를 삼키면 새로운 큐는 다음이 됩니다:

$$[x, a_1, \dots, a_m], [b_n, \dots, b_1].$$

원소를 빼면 새로운 큐는 다음이 됩니다:

$$[a_1, \dots, a_m], [b_{n-1}, \dots, b_1].$$

원소를 뺄 때, 때때로 L 리스트를 뒤집어서 R 로 გადა 놔야하겠습니다. 빈 큐는 $([], [])$ 이겠지요.

위의 세 함수를 K-로 정의하고 여러분이 작성한 K.run으로 테스트해 보기 바랍니다. □

Exercise 3 (30pts) “즐거운 고민”

저의 고민은 조카들이 모두 행복해 할 수 있도록 가장 저렴하게 선물을 준비하는 것입니다. 저의 조카들은 시샘이 많습니다. 매년 이맘때쯤이면 저는 조카들에게 선물을 한 꾸러미씩 나눠주는데, 받고나면 조카들끼리 다른 형제들이 받은 선물을 시샘하면서 서로 조르고 울고. 그래서 다시 정리해서 주면 또 만족스럽지 않아서 조르고 울고.

저는 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(싸우지 않을) 조건을 얘기하라고 합니다. 저는 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 CD ~~한~~ 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.”

구현을 위해서, 조카의 조건(require)은 다음과 같은 꼴로 표현된다고 정합시다:

“나는 최소한 ($cond_1$ 그리고 ... 그리고 $cond_k$)을 받아야 해요.”

nML 타입으로 정리하면 다음과 같습니다:

```
type require = id * (cond list)
and cond
    = Items of gift list          (* 선물들 *)
    | Same of id                 (* 어느 조카의 선물들 *)
```

```

    | Common of cond * cond          (* 두조건에 공통된 선물들 *)
    | Except of cond * gift list     (* 조건에서 어느 선물들은 빼고 *)
and gift = int                      (* 선물 번호 *)
and id = A | B | C | D | E         (* 조카 이름 *)

```

위의 다섯 조카들의 조건(require)을 받아서 최소의 선물쇼핑 리스트를 작성하는 shoppingList를 작성하기 바랍니다:

```
shoppingList: require list -> (id * gift list) list
```

결과는 조카마다 사주어야 할 선물들의 리스트입니다. 예를들어, 조카들의 조건이 다음과 같을때

- A: 최소한 $\{1, 2\}$ 하고 $\text{common}(B, C)$ 를 받아야.
- B: 최소한 $\text{common}(C, \{2, 3\})$ 를 받아야.
- C: 최소한 $\{1\}$ 하고 $(A \text{ except } \{3\})$ 를 받아야.

그러면 최소의 선물꾸러미들은 A에게 $\{1, 2\}$, B에게 $\{2\}$, C에게 $\{1, 2\}$ 이므로, shoppingList의 결과는

```
[(A, [1,2]), (B, [2]), (C, [1,2]), (D, nil), (E, nil)]
```

입니다. (조카마다 받는 선물꾸러미는 “집합”입니다, 즉, 한 선물 꾸러미에는 같은 선물이 두개이상 포함되지는 않습니다).

□