

Homework 3
SNU 4190.310, 2010 가을
이 광근
Program Due: 10/8, 24:00
Essay Due: 10/11, 15:30

Exercise 1 (30pts) “논술 에세이”

강의 홈페이지의 읽을거리 [Part I]에 매달린 글들을 읽고(비디오 포함) 리포트로 작성해서 제출합니다.

- 논술의 구성은: 읽은 내용 정리 30%, 읽고 느낀점 70%로.
- 각 단락은 두괄식으로. 두괄식이란, 단락의 결론을 단락의 첫 문장으로 가져오는 것을 말합니다. 단락 내용을 정리한 문장(topic sentence)이 단락의 첫 문장. 작성한 에세이 단락들의 첫 문장들만을 읽어도 논술의 흐름이 부드럽게 되는 지 확인.
- A4용지 총 10 페이지를 넘기지 말것.
- 반드시 컴퓨터로 출력해서 제출.
- 10/11(월) 수업시간에 제출. No delay acceptable.

□

Exercise 2 (40pts) “K- 실행기”

수업시간에 정의한 명령형 언어 K^{-1} 를 생각하자. 이번 숙제는 K- 프로그램 램을 의미정의대로 실행시키는 함수(interpreter)를 작성하는 것이다.

아래의 KMINUS 꼴을 가지는 모듈 K를 정의하라.

¹숙제를 위한 문법과 의미의 정확한 정의는 TA페이지 참고.

```

module type 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
            | SEQ of exp * exp          (* sequence *)
            | IF of exp * exp * exp    (* if-then-else *)
            | WHILE of exp * exp      (* while loop *)
            | LETV of id * exp * exp   (* variable binding *)
            | LETF of id * id list * exp * exp (* procedure binding *)
            | CALLV of id * exp list  (* call by value *)
            | CALLR of id * id list   (* call by reference *)
            | RECORD of (id * exp) list (* record construction *)
            | FIELD of exp * id       (* access record field *)
            | ASSIGN of id * exp      (* assign to variable *)
            | ASSIGNF of exp * id * exp (* assign to record field *)
            | 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 바깥 세상에 드러내게 됩니다. 실행중에 타입이 맞지 않는 프로그램이면 Error라는 예외상황을 발생시키고 프로그램 실행이 중단되어야 합니다. “Error”란 (if and only if) 정의된 의미 규칙으로는 그 프로그램의 의미가 정의될 수 없는 경우입니다. 입출력은 정수만 가능합니다. 출력은 정수를 화면에 뿌리고 “newline”을 프린트합니다. □

Exercise 3 (10pts) “K- 프로그래밍: 거스름 방법의 수”

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

우리나라에는 1원, 10원, 100원, 500원, 1000원, 5000원, 10000원, 50000원권이 있습니다. 주어진 액수의 거스름돈을 만들어 주는 방법의 수를 계산하는 함수 `numch`를 K-로 정의하라.

예를 들어 `numch(100)`은 11가지이다: 1원만 100개로 거스르는 경우 부터, 10원 1개와 1원 90개로 거스르기, ..., 10원만 10개로 거스르기.

힌트: 1원이하로만 거스르는 경우수는 1. 10원이하로만 거스르는 경우수는 1원이하로만 거스르는 경우수 + 10원을 하나이상 사용해서 10원이하로만 거스르는 경우수. 100원이하로만 거스르는 경우수는 10원이하로만 거스르는 경우수 + 100원을 하나이상 사용해서 100원이하로만 거스르는 경우수, 등등이다.

즉, 대략 다음과 같이 정의될 것이다. K--로 완성해서, 여러분이 작성한 `K.run`으로 테스트해 보기 바랍니다.

```
numch(n) = if n<10 then numch1(n)
           else if n<100 then numch10(n)
           ...

numch1(n) = 1
numch10(n) = if n<10 then numch1(n)
             else if ...
             else numch1(n) + numch10(n-10)
           ...
```

□

Exercise 4 (10pts) “K- 프로그래밍: compound data”

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

두갈래 나무구조(binary tree)를 만들고 쓸 수 있는 아래의 함수들을 정의하라:

```
leaf: int → tree (* a leaf tree *)
makeLtree: int × tree → tree (* a tree with only a left subtree *)
makeRtree: int × tree → tree (* a tree with only a right subtree *)
makeTree: int × tree × tree → tree (* a tree with both subtrees *)

isEmpty: tree → bool (* see if empty tree *)
rTree: tree → tree (* right subtree *)
lTree: tree → tree (* left subtree *)
nodeValue: tree → int (* node value *)
dft: tree → unit (* print node values in depth-first order *)
bft: tree → unit (* print node values in breath-first order *)
```

위의 함수들 만을 이용해서 나무 구조를 만들고 dft와 bft를 돌려서 제대로 된 순서로 출력되는 지를 확인하라.

참고로, 만든 실행기에 메모리 소모량을 측정하는 장치를 달고, 프로그램을 돌렸을 때 얼마만큼의 메모리를 소모하는 지를 재보자. 메모리 소모가 될 수 있으면 작도록 프로시저를 구현하도록 해 보자. □