Homework 5
CS520/KAIST Fall 2002
Kwangkeun Yi

due: 10/31 13:00 to me

Exercise 1 "Semantics Design of ImL (Imperative Language)"

Consider the following syntax that you all imperative programmers are used to:

$$
\begin{array}{rcl}
e & \rightarrow & z \mid \texttt{true} \mid \texttt{false} \mid \texttt{null} \\
& \mid & e \texttt{ + } e \mid e \texttt{ - } e \mid e \texttt{ * } e \\
& \mid & e \texttt{ = } e \mid e \texttt{ < } e \mid \texttt{not } e \\
& \mid & x \mid \texttt{read } x \mid \texttt{write } e \\
& \mid & x \texttt{ := } e \mid e \texttt{ ; } e \\
& \mid & \texttt{if } e \texttt{ then } e \texttt{ else } e \\
& \mid & \texttt{if } e \texttt{ then } e \\
& \mid & \texttt{while } e \texttt{ do } e \texttt{ end} \\
& \mid & \texttt{let var } x \texttt{ := } e \texttt{ in } e \texttt{ end} \\
& \mid & \texttt{let proc } f\texttt{(}x\texttt{) = } e \texttt{ in } e \texttt{ end} \\
& \mid & \texttt{call } f \texttt{ (}e\texttt{)}
\end{array}
$$

- Define yourself the semantics of the above language using one of the semantic formalisms discussed in class.
- There must be some semantic choices that you have to decide. Discuss about the characteristics (your choices) of the semantics.

Exercise 2 "SM5 Machine"

SM5 is an abstract machine. "SM" means "Stack Machine", "5" is because the machine has 5 components:

$$(S, M, E, C, K)$$

S is a stack (LIFO-access memory), M is a random-access memory, E is an environment, C is a command sequence, K is a thing-to-do sequence(called "continuation"): (v, x, l, z, b, w,and p respectively

denotes the elements of $Value, Var, Loc, Integer, Bool, Svalue, Proc$.)

$$
\begin{array}{lrcl}
S & \in & Stack & = & Svalue\ list \\
M & \in & Memory & = & Loc \to Value \\
E & \in & Environment & = & Var \to Loc + Proc \\
C & \in & Command & = & Cmd\ list \\
K & \in & Continuation & = & (Command \times Environment)\ list \\
v & \in & Value & = & Integer + Bool + Unit \\
x & \in & Var & & \\
l & \in & Loc & & \\
z & \in & Integer & & \\
b & \in & Bool & & \\
w & \in & Svalue & = & Value + Loc + Proc \qquad \text{(* stackable values *)} \\
p & \in & Proc & = & Var \times Command \times Environment \\
 & & Cmd & = & \{\texttt{push}\ v, \texttt{push}\ x, \texttt{push(}x\texttt{,}C\texttt{)}, \\
 & & & & \ \texttt{pop, store, load, jtr(}C\texttt{,}C\texttt{)}, \\
 & & & & \ \texttt{malloc, bind}\ x\texttt{, ubind, get, put, call,} \\
 & & & & \ \texttt{add, sub, mul, div, eq, less, not}\}
\end{array}
$$

The SM5 machine runs by the following transition

$$(S, M, E, C, K) \Rightarrow (S', M', E', C', K')$$

defined as:

$$
\begin{array}{rlllll}
 & (S, & M, & E, & \texttt{push}\ v :: C, & K) \\
\Rightarrow & (v :: S, & M, & E, & C, & K) \\
 & & & & & \\
 & (S, & M, & E, & \texttt{push}\ x :: C, & K) \\
\Rightarrow & (E(x) :: S, & M, & E, & C, & K) \\
 & & & & & \\
 & (S, & M, & E, & \texttt{push}\ (x, C') :: C, & K) \\
\Rightarrow & ((x, C', E) :: S, & M, & E, & C, & K) \\
 & & & & & \\
 & (w :: S, & M, & E, & \texttt{pop} :: C, & K) \\
\Rightarrow & (S, & M, & E, & C, & K) \\
 & & & & & \\
 & (l :: v :: S, & M, & E, & \texttt{store} :: C, & K) \\
\Rightarrow & (S, & M[v/l], & E, & C, & K) \\
 & & & & & \\
 & (l :: S, & M, & E, & \texttt{load} :: C, & K) \\
\Rightarrow & (M(l) :: S, & M, & E, & C, & K) \\
 & & & & & \\
 & (\mathit{true} :: S, & M, & E, & \texttt{jtr}(C_1, C_2) :: C, & K) \\
\Rightarrow & (S, & M, & E, & C_1 :: C, & K) \\
 & & & & & \\
 & (\mathit{false} :: S, & M, & E, & \texttt{jtr}(C_1, C_2) :: C, & K) \\
\Rightarrow & (S, & M, & E, & C_2 :: C, & K) \\
 & & & & & \\
 & (S, & M, & E, & \texttt{malloc} :: C, & K) \\
\Rightarrow & (l :: S, & M, & E, & C, & K) \qquad \text{new } l
\end{array}
$$

$$
\begin{array}{lllllll}
 & (w :: S, & M, & E, & \texttt{bind}\ x :: C, & K) & \\
\Rightarrow & (S, & M, & (x, w) :: E, & C, & K) & \\
 & & & & & & \\
 & (S, & M, & (x, w) :: E, & \texttt{ubind} :: C, & K) & \\
\Rightarrow & (S, & M, & E, & C, & K) & \\
 & & & & & & \\
 & (S, & M, & E, & \texttt{get} :: C, & K) & \\
\Rightarrow & (z :: S, & M, & E, & C, & K) & \text{read } z \text{ from outside} \\
 & & & & & & \\
 & (z :: S, & M, & E, & \texttt{put} :: C, & K) & \\
\Rightarrow & (S, & M, & E, & C, & K) & \text{print } z \text{ and newline} \\
 & & & & & & \\
 & (z_2 :: z_1 :: S, & M, & E, & \texttt{add} :: C, & K) & \\
\Rightarrow & ((z_1 + z_2) :: S, & M, & E, & C, & K) & \text{similar for } \texttt{sub,mul,div} \\
 & & & & & & \\
 & (v_2 :: v_1 :: S, & M, & E, & \texttt{eq} :: C, & K) & \\
\Rightarrow & ((v_1 = v_2) :: S, & M, & E, & C, & K) & \\
 & & & & & & \\
 & (z_2 :: z_1 :: S, & M, & E, & \texttt{less} :: C, & K) & \\
\Rightarrow & ((z_1 < z_2) :: S, & M, & E, & C, & K) & \\
 & & & & & & \\
 & (b :: S, & M, & E, & \texttt{not} :: C, & K) & \\
\Rightarrow & (\neg b :: S, & M, & E, & C, & K) & \\
 & & & & & & \\
 & (l :: v :: (x, C', E') :: S, & M, & E, & \texttt{call} :: C, & K) & \\
\Rightarrow & (S, & M[v/l], & (x, l) :: E', & C', & (C, E) :: K) & \\
 & & & & & & \\
 & (S, & M, & E, & \mathit{empty}, & (C, E') :: K) & \\
\Rightarrow & (S, & M, & E', & C, & K) &
\end{array}
$$

Semantics of an SM5 program C is defined to be the transition sequence of the SM5 machine:

$$(\mathit{empty}, \mathit{empty}, \mathit{empty}, C, \mathit{empty}) \Rightarrow \cdots \Rightarrow \cdots$$

Define a correct compilation rules from ImL (Exercise 1) to C.