Meaning Function of Clite (cont.)

- We now know that the meaning function of assignment is defined as
  \[ M(\text{assignment}, \text{oldstate}) = \text{newstate} \]
  - Read as the meaning of this assignment statement in this state is this new state.

- An if statement changes the state as well. So use the same methodology, we can define the meaning function for if statement as
  \[ M(\text{ifstatement}, \text{oldstate}) = \text{newstate} \]

- We can also use the same methodology to define the meaning function for a block.
  \[ M(\text{blockstatement}, \text{oldstate}) = \text{newstate} \]

- Actually, all Clite statements are just functions from \textit{state1} \rightarrow \textit{state2}

- We will combine all these functions into one function that takes two parameters.
  \[ M : \text{Statement} \times \text{State} \rightarrow \text{State} \]

- The meaning of a statement is a new state that depends both on the old state and the statement being executed.

- Follow the same methodology, we can define the function for Program and Expression:
  \[ M : \text{Program} \rightarrow \text{State} \]
  \[ M : \text{Expression} \times \text{State} \rightarrow \text{Value} \]

- A value could be any kind of r-value such as an integer, float, or any other mathematically precise object.

- Let’s start implementing M for Clite.

- Note that, since we are thinking of expressions and statements in terms of the abs tree, and since the meanings of each programming component (e.g., expression, assignment, statement, etc.) depend on the meaning of the subtrees, guess what we will have when we implement these functions? [Lots of recursion]
Program Meaning Function

- The meaning function of Program can be implemented into this.

\[
M : Program \rightarrow State
\]

\[
M(\text{Program } p) = M(\text{p.body, InitialState}(!\text{p.decpart}))
\]

- If we have a Program \( p \), the implemented function should take two parameters, the body of \( p \) and the initial state of \( p \)'s decpart.

- In this implementation, the InitialState is an auxiliary function that creates a state containing all the variables declared in \( p \).decpart paired with their default values.

- For example, if \( \text{int } x = 0, y = 0; \) is \( p \).decpart in C, then \( \text{initState}(!\text{p.decpart}) = \{<x,0>, <y,0>\} \).

- Note:
  - The "meaning" of a Program is the final state of the computer's memory when the program halts.
  - The \( p \).body part uses recursion. Depends on the statement type in \( p \).body, it will be implemented differently. For example, expression will have a different semantics than assignment.

InitialState in Programming Languages

- The InitialState function must be clearly defined for a language in order for programs to have meaning. [show initialState.c, initialState.java, initialState2.java]

```c
/**
 * initialState.c
 * Ying Li
 */
#include <stdio.h>
int main () {
    int sum;
    for (int i = 0; i < 10; i++) {
        sum += 5;
    }
    printf("sum = %d \n", sum);
    return 0;
}
```

- C does not initialize the memory space. If not explicit initialization, the initial value of a variable is the random value stored in that memory space. (So, initialState.c gives different results every time you run it)
Java requires the variables being initialized before using. So initialState.java gives an error for no initialization. But if the variables are fields in a class, even without a constructor, initialState2.java won’t complain, since a default constructor will be called. Default constructor will initialize the variables to the default values of their types, e.g. int is 0, float is 0.0 and string is "".