

CSE 755, Assignment #2
Due: Jan. 31, '12 (No late assignments)

1. (8 points). Consider the *IMP* language whose translational semantics we defined in class (notes: pages 17–20). Suppose we revise IMP to be a block structured language (but no procedures or functions). Here is the (partial) BNF for the language:

```
<prog> ::= <block>
<stmt> ::= <assign> | <if> | <loop> | <stmt> <stmt> | <block>
<block> ::= begin <ds> <stmt> end;
  <ds> ::= <decl> | <decl> <ds>
  <decl> ::= int <id-list>;
<assign>, <if>, <loop> are as before.
```

So a `<stmt>` can be a `<block>` which is how you get nested blocks.

How would you revise the translational semantics of IMP to account for these changes? Note that you can no longer use the names of variables as we did because in the same program we may have, in two different blocks or possibly an outer block and a nested block, two different variables declared that are both called *XYZ*, so if you were to use *XYZ* in the assembly language code, it would be ambiguous. Instead, you would have to introduce a sort of symbol table, assign addresses (or something) to the variables, use those addresses in the code you produce. And in the attribute grammar, you would have to set up suitable mechanisms to achieve all this. It is these mechanisms, as well as how you would use them to produce the translational semantics, that you have to describe. (You will also have to make some minor changes in the assembly language instructions that we used; use whatever conventions you think are appropriate for this purpose, but make sure you describe them. Also, your conventions must be reasonable, so you can't assume, for example, that the assembly language will take care of the block structure for you!)

2. (6 points). Write a Lisp function that will take two arguments *L1*, *L2*, both of which are lists of atoms, and return a list (of atoms) obtained by appending together *L1* and *L2*. So, given (1 3 5) and (2 4 6 7) as the arguments, your function should return (1 3 5 2 4 6 7) as the result.
3. (6 points). Write a Lisp function that takes an arbitrary s-expression *S* as argument and returns *T* or *NIL* depending on whether or not there is *any* atom that appears more than once in *S*. Note that *S* need not be a list of atoms; it can be any s-expression. So the question is, is there some atom, say *XYZ*, that appears more than once in *S*. If there is such an atom, return *T*, else return *NIL*.

Note: The first mid-term will be on Thursday, Feb. 2. Topics will be everything we discuss in class by then.