Firstly, we need to allow the C expression **‘int variable’** to compile, and to assume the 0 value when putting it into Lisp (the Lisp interpreter does not allow the assignment operation without a second parameter). To do this we created two new non-terminals: definition and recursion_def.
In definition -> IDENTIF (IDENTIF represents our variable for the time being), we just print **‘setq variable 0’**.
recursion_def will be used later on.

For the second part we need to allow expressions of the sort: **‘int variable = number’.** For that we add the operation definition -> IDENTIF = NUMBER, and we simply print **‘setq variable number’.**

To accept multiple definitions of variables separated by a comma we use another non-terminal recursion_def, in which we implement the necessary recursion for chained variable assignments. The recursion is as follows:
recursion_def -> definition ‘,’ recursion_def | definition.

To allow statements including the main function we create a new non-terminal, code, which will consist of all the instructions inside the main function. We add a new derivation in our axiom to detect if we are declaring a function, this is: axiom -> MAIN ‘(‘ ‘)’ ‘{’ code ‘}’ , which prints: **‘defun main () code**. Which is the syntax the Lisp interpreter allows. For code we allow exactly the same operations that we did earlier.