The change I made was to <assign> and <print>
<assign> can now take string values and store them in variables
And <print> can take those variables as arguments
11.5: <p-arg> -> ID
13.5 <assign> -> ID “=” STRING

Static Semantics
11.5 ID.id = <p-arg>.id
The dynamic semantics for these new changes are as such
densem(“print” ID,(Θ,i,p)) = (Θ,i,append(p,Θ(ID))

densem(“get” ID,(Θ,i,p)) = (Θ’,i’,p)
Where
(x,i’) = get(clean(i))
Θ’(n) = if n = ID then x else Θ(n)
//we do get instead of getInt to say we are just storing a typeless variable

I decided to implement a symbol table and I ran the subprogram for adding to the symbol table within the parsing part of the compiler. I would like to rework my code so that adding to the symbol table is instead done during the lexing.
I also decided to terminate on any syntax error, since a lack of proper formatting might have affected any subsequent calculation of other expressions and statements.

I could not implement any loops, as it would have been too difficult to implement a way to retain the previous tokens for the <expr> and <stmt_list> parts of the loop. I think a modification to the function I used to backtrack to a previous token might have been useful for implementation of a while loop.