Predict sets for LL grammar

Predict( <prog> -> <decl\_list> Scope EOL <stat\_list> End Scope EOF) = { Declare, Function, Scope, Dim, Static,EOL }  
Predict( <decl\_list> -> <var\_decl> <decl\_list> ) = { Dim, Static }  
Predict( <decl\_list> -> <func\_decl> <decl\_list> ) = { Declare }  
Predict( <decl\_list> -> <func\_def> <decl\_list> ) = { Function }  
Predict( <decl\_list> -> <empty\_statement> <decl\_list> ) = { EOL }  
Predict( <decl\_list> -> ε ) = { Scope }  
Predict( <func\_decl> -> Declare Function id ( <param\_list> ) As <type> EOL ) = { Declare }  
Predict( <func\_def> -> Function id ( <param\_list> ) As <type> EOL <stat\_list> End Function EOL ) = { Function }  
Predict( <var\_decl> -> Dim <opt\_scope\_modifier> id As <type> <opt\_initialiser> EOL ) = { Dim }  
Predict( <var\_decl> -> Static id as <type> <opt\_initialiser> EOL ) = { Static }  
Predict( <param\_list> -> <param> <param\_list\_cont> ) = { id }  
Predict( <param\_list> -> ε ) = { ) }  
Predict( <param> -> id As <type> ) = { id }  
Predict( <param\_list\_cont> -> , <param> <param\_list\_cont> ) = { , }  
Predict( <param\_list\_cont> -> ε ) = { ) }  
Predict( <opt\_scope\_modifier> -> Shared ) = { Shared }  
Predict( <opt\_scope\_modifier> -> ε ) = { id }  
Predict( <opt\_initialiser> -> = <expr> ) = { = }  
Predict( <opt\_initialiser> -> ε ) = { EOL }  
Predict( <type> -> Integer ) = { Integer }  
Predict( <type> -> Double ) = { Double }  
Predict( <type> -> String ) = { String }  
Predict( <stat\_list> -> <statement> <stat\_list> ) = { id, Input, Print, Scope, If, For, Do, Return, Dim, Static, continue, exit EOL }  
Predict( <stat\_list> -> ε ) = { End, elseif, else, next, loop}  
Predict( <statement> -> <assignment> ) = { id }  
Predict( <statement> -> <read\_statement> ) = { Input }  
Predict( <statement> -> <print\_statement> ) = { Print }  
Predict( <statement> -> <scope\_statement> ) = { Scope }  
Predict( <statement> -> <selection\_statement> ) = { If }  
Predict( <statement> -> <iteration\_statement> ) = { Do, For }  
Predict( <statement> -> <return\_statement> ) = { return }  
Predict( <statement> -> <empty\_statement> ) = { EOL }  
Predict( <statement> -> <var\_decl> ) = { Dim, Static }  
Predict( <statement> -> <iteration\_control\_statement> ) = { continue, exit }  
Predict( <assignment> -> id = <expr> EOL ) = { id }  
Predict( <read\_statement> -> Input id EOL ) = { Input }  
Predict( <print\_statement> -> print <expr> ; <expr\_list> EOL ) = { print }  
Predict( <expr\_list> -> <expr> ; <expr\_list> ) = { id, ( , int\_literal, double\_literal, string\_literal, length, asc, substr, chr }  
Predict( <expr\_list> -> ε ) = { EOL }  
Predict( <scope\_statement> -> Scope EOL <stat\_list> End Scope EOL ) = { Scope }  
Predict( <selection\_statement> -> If <expr> then EOL <stat\_list> <alternative\_statement> End If EOL ) = { If }  
Predict( <alternative\_statement> -> elseif <expr> then EOL <stat\_list> <alternative statement> ) = { elseif }  
Predict( <alternative\_statement> -> else EOL <stat\_list> ) = { else }  
Predict( <alternative\_statement> -> ε ) = { End }  
Predict( <iteration\_statement> -> for id <opt\_type\_decl> = <expr> to <expr> <opt\_step> EOL <stat\_list> Next <opt\_id> EOL ) = { for }  
Predict( <opt\_type\_decl> -> as <type> ) = { as }  
Predict( <opt\_type\_decl> -> ε ) = { = }  
Predict( <opt\_step> -> step <expr> ) = { step }  
Predict( <opt\_step> -> ε ) = { EOL }  
Predict( <opt\_id> -> id ) = { id }  
Predict( <opt\_id> -> ε ) = { EOL }  
Predict( <iteration\_statement> -> do <do\_cycle> EOL ) = { do }  
Predict( <do\_cycle> -> EOL <stat\_list> Loop <opt\_cond> ) = { EOL }  
Predict( <do\_cycle> -> <cond> EOL <stat\_list> Loop ) = { until, while }  
Predict( <opt\_cond> -> <cond> ) = { until, while }  
Predict( <opt\_cond> -> ε ) = { EOL }  
Predict( <cond> -> <do\_mode> <expr> ) = { until, while }  
Predict( <do\_mode> -> until ) = { until }  
Predict( <do\_mode> -> while ) = { while }  
Predict( <return\_statement> -> return <expression> EOL ) = { return }  
Predict( <iteration\_control\_statement> -> <control\_statement> <cycle\_type> <cycle\_type\_list> EOL ) = { Exit, Continue }  
Predict( <control\_statement> -> Exit ) = { Exit }  
Predict( <control\_statement> -> Continue ) = { Continue }  
Predict( <cycle\_type\_list> -> , <cycle\_type> <cycle\_type\_list> ) = { , }  
Predict( <cycle\_type\_list> -> ε ) = { EOL }  
Predict( <cycle\_type> -> for ) = { for }  
Predict( <cycle\_type> -> do ) = { do }  
Predict( <empty\_statement> -> EOL ) = { EOL }  
Predict( <func\_call> -> ( <arg\_list> ) ) = { ) }  
Predict( <arg\_list> -> <arg> <more\_args\_list> ) = { (, id, int\_literal, double\_literal, string\_literal, length, asc, chr, substr }  
Predict( <arg\_list> -> ε ) = { ) }  
Predict( <more\_args\_list> -> , <arg> <more\_args\_list> ) = { , }  
Predict( <more\_args\_list> -> ε ) = { ) }  
Predict( <arg> -> <expr> ) = { (, id, int\_literal, double\_literal, string\_literal, length, asc, chr, substr }  
Predict( <built\_in\_func\_call> -> length <func\_call> ) = { length }  
Predict( <built\_in\_func\_call> -> substr <func\_call> ) = { substr }  
Predict( <built\_in\_func\_call> -> asc <func\_call> ) = { asc }  
Predict( <built\_in\_func\_call> -> chr <func\_call> ) = { chr }