This grammar is lightly adapted from the provided grammar. I refactored both declaration-list and param to convert this from a LL3 grammar to a LL1 grammar. (Before, the rule was param := type-specifier ID [] | type-specifier ID ε. By refactoring, we no longer have too look ahead three items to tokens determine which rule applies here).

1. program := declaration-list  
      | ε  
 2. declaration-list := declaration-list type-specifier ID declaration  
      | type-specifier ID declaration  
 3. declaration := var-declaration  
      | fun-declaration  
 4. var-declaration := ;  
      | [ NUM ] ;  
 5. type-specifier := int  
      | void  
 6. fun-declaration := ( params ) compound-stmt  
 7. params := param-list  
      | void  
      | empty  
 8. param-list := param-list , type-specifier ID param  
      | type-specifier ID param  
 9. param := []  
      | ε  
 10. compound-stmt := { local-declarations statement-list }  
 11. local-declarations := local-declarations var-declaration  
      | var-declaration  
 12. statement-list := statement-list statement  
      | statement  
 13. statement := expression-stmt  
      | compound-stmt  
      | selection-stmt  
      | iteration-stmt  
      | return-stmt  
 14. expression-stmt := expression ;  
      | ;  
 15. selection-stmt := if (simple- expression ) statement  
      | if ( simple-expression ) statement else statement  
 16. iteration-stmt := while ( expression ) statement  
 17. return-stmt := return ;  
      | return expression ;  
 18. expression  := var = expression  
      | simple-expression  
 19. var := ID | ID [ simple-expression ]  
 20. simple-expression := additive-expression relop additive-expression  
      | additive-expression  
 21. relop := <=  
      | <  
      | >  
      | >=  
      | ==  
      | !=  
 22. additive-expression := additive-expression addop term  
      | term  
 23. add-op := +  
      | -  
 24. term := term mulop factor  
      | factor  
 25. mulop := \*  
      | /  
 26. factor := ( simple-expression )  
      | var  
      | call  
      | NUM  
 27. call := ID ( args )  
 28. args := arg-list  
      | ε  
 29. arg-list := arg-list , expression  
      | expression