BNF Grammar

program --> declaration program | *epsilon*

declaration --> **void id_const** fun-dec-tail | nonvoid-specifier **id_const** de-tail

nonvoid-specifier --> **int** | **bool**

id_const -> **ID**

dec-tail --> var-dec-tail | fun-dec-tail

var-dec-tail --> **[**add-exp**]** var-dec-tail' ; | var-dec-tail' ;

var-dec-tail' --> , var-name var-dec-tail' | *epsilon*

var-name --> **id_const** var-name'

var-name' --> **[**add-exp**]** | *epsilon*

fun-dec-tail --> **(**params**)** compound-stmt

params --> param params' | **void**

params' --> , param params' | *epsilon*

param --> **ref** nonvoid-specifier **id_const** param | nonvoid-specifier **id_const** param'

param' --> **[]** | *epsilon*

statement --> id-stmt | compound-stmt | if-stmt | loop-stmt | exit-stmt | continue-stmt | return-stmt
| null-stmt

id-stmt --> **id_const** id-stmt-tail

id-stmt-tail --> assign-stmt-tail | call-stmt-tail

assign-stmt-tail --> **[**add-exp**] :=** expression **;** | **:=** expression **;**

call-stmt-tail --> call-tail **;**

call-tail --> **(** call-tail' **)**

call-tail' --> arguments | *epsilon*

arguments --> expression arguments'

arguments' --> **,** expression arguments' | *epsilon*

compound-stmt --> **{** compound-stmt' compound-stmt'' **}**

compound-stmt' --> nonvoid-specifier **id_const** var-dec-tail compound-stmt-' | *epsilon*

compound-stmt'' --> statement compound-stmt'''

compound-stmt''' --> statement compound-stmt''' | *epsilon*

if-stmt --> **if (** expression **)** statement if-stmt'

if-stmt' --> **else** statement | *epsilon*

loop-stmt --> **loop** statement loop-stmt' **end ;**

loop-stmt' -->statement loop-stmt' | *epsilon*

exit-stmt --> **exit ;**

continue-stmt --> **continue ;**

return-stmt --> **return** return-stmt' ;

return-stmt' --> expression | *epsilon*

null-stmt --> **;**

expression --> add-expr expression'

expression' --> relop add-exp | *epsilon*

add-exp --> uminus term add-exp' | term add-exp'

add-exp' --> addop term add-exp' | *epsilon*

term --> factor term'

term' --> multop factor term' | *epsilon*

factor --> nid-factor | id-factor

nid-factor --> **not** factor | **(** expression **)** | **num** | **blit**

id-factor --> **id_const** id-tail

id-tail --> var-tail | call-tail

var-tail --> **[**add-exp**]** | *epsilon*

relop --> <= | < | > | >= | = | /=

addop --> + | **-** | **or** | **orelse**

multop --> ***** | / | **mod** | **and** | **andthen**

uminus --> **-**