**Grammer**

PROG -> GLOBAL\_VARS FUNC\_PREDEFS FUNC\_FULL\_DEFS

GLOBAL\_VARS -> GLOBAL\_VARS VAR\_DEC | VAR\_DEC /\* declarations of global variables \*/

VAR\_DEC -> TYPE id ; | TYPE id [ DIM\_SIZES ] ; /\* allow multi‐dimensional arrays \*/

TYPE -> **int** | **float** /\* variables can be only of these types \*/

DIM\_SIZES -> int\_num | int\_num , DIM\_SIZES /\* list of sizes in each of the dimensions \*/

FUNC\_PREDEFS -> FUNC\_PREDEFS FUNC\_PROTOTYPE ; | FUNC\_PROTOTYPE ;

FUNC\_PROTOTYPE -> RETURNED\_TYPE id ( PARAMS )

FUNC\_FULL\_DEFS -> FUNC\_WITH\_BODY FUNC\_FULL\_DEFS | FUNC\_WITH\_BODY

FUNC\_WITH\_BODY -> FUNC\_PROTOTYPE COMP\_STMT

RETURNED\_TYPE -> TYPE | **void**

PARAMS -> PARAM\_LIST | ɛ /\* function can be without parameters \*/

PARAM\_LIST -> PARAM\_LIST , PARAM | PARAM

PARAM -> TYPE id | TYPE id [ DIM\_SIZES ]

COMP\_STMT -> { VAR\_DEC\_LIST STMT\_LIST } /\* if VAR\_DEC\_LIST is non‐empty, then

COMP\_STM is in fact a block that

contains declarations of local variables.

Otherwise it is just a grouped series of

statements \*/

VAR\_DEC\_LIST -> VAR\_DEC\_LIST VAR\_DEC | ɛ

STMT\_LIST -> STMT\_LIST ; STMT | STMT

STMT -> VAR = EXPR | COMP\_STMT | IF\_STMT | CALL | RETURN\_STMT

/\* note that in the assignment, the left hand

side can be either a simple variable, or an array

element – see definition of VAR below \*/

IF\_STMT -> **if** ( CONDITION ) STMT /\* note that STMT can be a COMP\_STMT, thus

allowing execution of any amount of

statements when condition is True \*/

CALL -> id ( ARGS )

ARGS -> ARG\_LIST | ɛ

ARG\_LIST -> ARG\_LIST , EXPR | EXPR

RETURN\_STMT -> **return** | **return** EXPR

VAR -> id | id [ EXPR\_LIST ] /\* to allow access to multi‐dimensional arrays \*/

EXPR\_LIST -> EXPR , EXPR\_LIST | EXPR

CONDITION -> EXPR rel\_op EXPR

EXPR -> EXPR + TERM | TERM

TERM -> TERM \* FACTOR | FACTOR

FACTOR -> VAR | CALL | int\_num | float\_num | ( EXPR )

**Grammer without left recursion**

PROG -> GLOBAL\_VARS FUNC\_PREDEFS FUNC\_FULL\_DEFS

GLOBAL\_VARS -> VAR\_DEC GLOBAL\_VARS'

GLOBAL\_VARS' -> VAR\_DEC GLOBAL\_VARS' | ɛ

VAR\_DEC -> TYPE id ; | TYPE id [ DIM\_SIZES ] ;

TYPE -> int | float

DIM\_SIZES -> int\_num | int\_num , DIM\_SIZES

FUNC\_PREDEFS -> FUNC\_PROTOTYPE ; FUNC\_PREDEFS'

FUNC\_PREDEFS' -> FUNC\_PROTOTYPE ; FUNC\_PREDEFS' | ɛ

FUNC\_PROTOTYPE -> RETURNED\_TYPE id ( PARAMS )

FUNC\_FULL\_DEFS -> FUNC\_WITH\_BODY FUNC\_FULL\_DEFS | FUNC\_WITH\_BODY

FUNC\_WITH\_BODY -> FUNC\_PROTOTYPE COMP\_STMT

RETURNED\_TYPE -> TYPE | void

PARAMS -> PARAM\_LIST | ɛ

PARAM\_LIST -> PARAM PARAM\_LIST'

PARAM\_LIST' -> , PARAM PARAM\_LIST' | ɛ

PARAM -> TYPE id | TYPE id [ DIM\_SIZES ]

COMP\_STMT -> { VAR\_DEC\_LIST STMT\_LIST }

VAR\_DEC\_LIST -> VAR\_DEC\_LIST'

VAR\_DEC\_LIST' -> VAR\_DEC VAR\_DEC\_LIST' | ɛ

STMT\_LIST -> STMT STMT\_LIST'

STMT\_LIST' -> ; STMT STMT\_LIST' | ɛ

STMT -> VAR = EXPR | COMP\_STMT | IF\_STMT | CALL | RETURN\_STMT

IF\_STMT -> if ( CONDITION ) STMT

CALL -> id ( ARGS )

ARGS -> ARG\_LIST | ɛ

ARG\_LIST -> EXPR ARG\_LIST'

ARG\_LIST' -> , EXPR ARG\_LIST' | ɛ

RETURN\_STMT -> return | return EXPR

VAR -> id | id [ EXPR\_LIST ]

EXPR\_LIST -> EXPR , EXPR\_LIST | EXPR

CONDITION -> EXPR rel\_op EXPR

EXPR -> TERM EXPR'

EXPR' -> + TERM EXPR' | ɛ

TERM -> FACTOR TERM'

TERM' -> \* FACTOR TERM' | ɛ

FACTOR -> VAR | CALL | int\_num | float\_num | ( EXPR )

**Grammer without common left prefixes**

PROG -> GLOBAL\_VARS FUNC\_PREDEFS FUNC\_FULL\_DEFS

GLOBAL\_VARS -> VAR\_DEC GLOBAL\_VARS'

GLOBAL\_VARS' -> VAR\_DEC GLOBAL\_VARS' | ɛ

VAR\_DEC -> TYPE id VAR\_DEC'

VAR\_DEC' -> ; | [ DIM\_SIZES ] ;

TYPE -> int | float

DIM\_SIZES -> int\_num DIM\_SIZES'

DIM\_SIZES' -> , int\_num DIM\_SIZES' | ɛ

FUNC\_PREDEFS -> FUNC\_PROTOTYPE ; FUNC\_PREDEFS'

FUNC\_PREDEFS' -> FUNC\_PROTOTYPE ; FUNC\_PREDEFS' | ɛ

FUNC\_PROTOTYPE -> RETURNED\_TYPE id ( PARAMS )

FUNC\_FULL\_DEFS -> FUNC\_WITH\_BODY FUNC\_FULL\_DEFS'

FUNC\_FULL\_DEFS' -> FUNC\_WITH\_BODY FUNC\_FULL\_DEFS' | ɛ

FUNC\_WITH\_BODY -> FUNC\_PROTOTYPE COMP\_STMT

RETURNED\_TYPE -> TYPE | void

PARAMS -> PARAM\_LIST | ɛ

PARAM\_LIST -> PARAM PARAM\_LIST'

PARAM\_LIST' -> , PARAM PARAM\_LIST' | ɛ

PARAM -> TYPE id PARAM'

PARAM' -> [ DIM\_SIZES ] | ɛ

COMP\_STMT -> { VAR\_DEC\_LIST STMT\_LIST }

VAR\_DEC\_LIST -> VAR\_DEC\_LIST'

VAR\_DEC\_LIST' -> VAR\_DEC VAR\_DEC\_LIST' | ɛ

STMT\_LIST -> STMT STMT\_LIST'

STMT\_LIST' -> ; STMT STMT\_LIST' | ɛ

STMT -> id VAR` = EXPR | COMP\_STMT | IF\_STMT | CALL | RETURN\_STMT

IF\_STMT -> if ( CONDITION ) STMT

CALL -> id ( ARGS )

ARGS -> ARG\_LIST | ɛ

ARG\_LIST -> EXPR ARG\_LIST'

ARG\_LIST' -> , EXPR ARG\_LIST' | ɛ

RETURN\_STMT -> return RETURN\_STMT'

RETURN\_STMT' -> EXPR | ɛ

VAR -> id VAR'

VAR' -> [ EXPR\_LIST ] | ɛ

EXPR\_LIST -> EXPR EXPR\_LIST'

EXPR\_LIST' -> , EXPR EXPR\_LIST' | ɛ

CONDITION -> EXPR rel\_op EXPR

EXPR -> TERM EXPR'

EXPR' -> + TERM EXPR' | ɛ

TERM -> FACTOR TERM'

TERM' -> \* FACTOR TERM' | ɛ

//FACTOR -> VAR | CALL | int\_num | float\_num | ( EXPR )

FACTOR -> id MOMO | int\_num | float\_num | ( EXPR )

MOMO-> VAR` | ( ARGS )