myC-

<program> ::= int main() <block>

<block> ::= { <block statements>} | {}

<block statements> ::= <block statement>; | <block statement>; <block statements>

<block statement> ::= <local variable declaration> | <statement> | <function declaration>

<local variable declaration > ::= <type> <variable declarators>

<variable declarators> ::= <variable declarator> | <variable declarators>, <variable declarator>

<variable declarator id> ::= <identifier> | <variable declarator id>[]

<statement> ::= <assign statement> | <if then statement> | <while statement> |<empty statement> |<return statement>

<empty statement> ::= ;

<assign statement> ::= <expression statement>

<expression statement> ::= <expression>;

<while statement> = while (expression) do <statement>

<if then statement> ::= if(<expression>) <statement>

<return statement> ::= return <expression>; | return;

<expression> ::= <assignment expression>

| <expression>, <assignment-expression>

|‘!’ <expression>

|<arithmetic expression>

| <boolean expression>

|<function invocation>

|<string expression>

<assignment expression> ::= <lhs> = <expression>

<lhs> ::= <identifier> | <get from array>

<get from array> ::= <identifier> [<expression>]

<string expression> ::= <exp>.<string chars> | <string type>

<arithmetic expression> ::= <arithmetic expression> <addop> <multip expression>

| <multip expression>

<multip expression> ::= <multip expression> <multop> <factor> | factor

<factor> ::= ( <arithmetic expression> ) | <variable> | -<factor> | numeral

<boolean expression> ::= <boolean literal> || <multip bexp> | <multip bexp>

<multip bexp> ::= <mult bexp> && <root bexp> | <root bexp>

<root bexp> ::= ! <root bexp> | <arithmetic expression> <relop> <arithmetic expression>

<relop> ::= < | <= | >= | > | ==

<addop> ::= + | -

<multop> ::= \* | / | %

<variable> ::= <identifier>

*It is assumed that numerals (float, int), characters (char), identifier, boolean (true/false) are given by scanner.*

<type> :: = <primitive type>

<array type> ::= <primitive type><array dim>

<array dim> :: = [] | <array dim> []

<primitive type> ::= <numeric type> | <boolean literal> | <string type>

<numeric type> ::= <int type> | <float type> | <char literal>

<string type> ::= “ <string chars> ”

<string chars> ::= <char literal> | <string chars>

<function name> ::= <identifier>

<function declaration> ::= <type> <function name> ‘(‘ <param list> ‘)’ <block>

| void <identifier> ‘(‘ <param list> ‘)’ <block>

<function invocation> ::= <function name> ( <param list> )

<param list> ::= expression> | <param list>, <expression>