Grammar for the Exp programming language

Version B

program → (function)* main

function → **def name (** (parameters)? **)** (**int**)? **{** (statement)* **}**

parameters → **name** (**,** **name**)*

main → (statement)+

statement → st_print | st_attrib | st_if | st_while | st_break | st_continue
| st_array_new | st_array_push | st_array_set | st_call | st_return

st_print → **print (** expression (**,** expression)* **)**

st_attrib → **name =** expression

st_if → **if** comparison **{** (statement)+ (**}** **else** **{** (statement)+ **}**)? **}**

st_while → **while** comparison **{** (statement)+ **}**

st_break → **break**

st_continue → **continue**

st_array_new → **name = [ ]**

st_array_push → **name . push (** expression **)**

st_array_set → **name [** expression **] =** expression

st_call → **name (** (arguments)? **)**

arguments → expression (**,** expression)*

st_return → **return** expression

comparison → expression (**==** | **!=** | **<** | **<=** | **>** | **>=**) expression

expression → term ((**+** | **-**) term)*

term → factor ((* | / | %) factor)*

factor → **number** | **string** | **(** expression **)** | **name** | **read_int ( )** | **read_str ( )**
| **name . length** | **name [** expression **]** | **name (** (arguments)? **)**

---

tokens: **+ - * / % ( ) = , { } == != < <= > >= [ ] .**
number string name read_int read_str
print if else while break continue push length def int return

bytecode instructions:

+1	**ldc** *integer or "string"*	
-1	**iadd**	
-1	**isub**	
-1	**imul**	
-1	**idiv**	
-1	**irem**	
-1	**istore** *index*	
+1	**iload** *index*	
-1	**astore** *index*	
+1	**aload** *index*	
+1	**getstatic**	
-2	**invokevirtual .../print(I)V**	
-2	**invokevirtual .../print(Ljava/lang/String;)V**	
-1	**invokevirtual .../println()V**	
+1	**invokestatic .../readInt()I**	
-2	**if_icmp??** *label*	**??** → **eq** | **ne** | **lt** | **le** | **gt** | **ge**
0	**goto** *label*	
+1	**new Array**	
+1	**dup**	
-1	**invokespecial Array/<init>()V**	
-2	**invokevirtual Array/push(I)V**	
0	**invokevirtual Array/length()I**	
0	**invokevirtual Array/string()Ljava/lang/String;**	
-3	**invokevirtual Array/set(II)V**	
-1	**invokevirtual Array/get(I)I**	
0	**invokestatic Test/*function*()V**	
-*n*	**invokestatic Test/*function*(I..I)V**	*n* is the number of parameters
-*n*+1	**invokestatic Test/*function*(I..I)I**	*n* is the number of parameters