Grammar #1:
<sign> = ‘+’ | ‘-’
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<letter> ::= A | B | C | D | E | F
<hex> ::= <sign> ‘0x’? <digit> | <letter> | <hex> <digit> | <hex> <letter>
<Addition> ::= <hex> + <hex>
<Subtraction> ::= <hex> - <hex>

Grammar #2:

<program> ::= <stmt-list>
<stmt-list> ::= <stmt> | <stmt-list>
<stmt> ::= <identifier> = <expr>
<variable> ::= “VAR” <identifier>
<identifier> ::= <id character> | <identifier> <id character> | <identifier> <decimal digit>
<id character> ::= <letter>
<letter> ::= A|B|C ... |Z|a|b|c ... |z
<decimal digit> ::= 0|1|2|3|4|5|6|7|8|9
<integer> ::= ['-'] <decimal digit> | <integer> <decimal digit>
<string> ::=‘ " ’ <characters> ‘ ." ’ | ‘’
<characters> ::= <character> | <characters> <character>
<character> ::= ‘’ | {any EBCDIC character }
<expr> ::= <term>| < term><operator>< term>|<string >
<term> ::= <integer>| <decimal digit>| <identifier>|<letter>
<operator> ::= +|-|*|/|**
<addition> ::= < term>+<term>
<subtraction > ::= < term>-<term>
<multiplication >: := < term>*<term>
<division > ::= < term>/<term>
<exponentiation >. ::= < term>**<term>
<print> ::= PRINT <OBJ>
<OBJ> ::= <variable>|<string>|<integer>

RegEx:
1.
^VAR\s[a-zA-Z]+[a-zA-Z0-9]*$

2.
-?[0-9]+

3.
^“[a-zA-Z\s.]*”$

4.
[a-zA-Z]+[a-zA-Z0-9]*=[a-zA-Z]|[a-zA-Z0-9]*|"[a-zA-Z\s.]*"|-?[0-9]+

5.
To match the five operators
[\+\-\/]|[*]{1,2}

To match the operands(variables/numeric values) with the five operators
/([a-zA-Z]+[a-zA-Z0-9]*|[0-9])([\+\-\/]|[*]{1,2})([a-zA-Z]+[a-zA-Z0-9]*|[0-9])/s

6.
To match the PRINT keyword
^PRINT

To match the object(variable, numeric value, string) after PRINT keyword
^PRINT\s("[a-zA-Z\s.]*"|VAR\s[a-zA-Z]+[a-zA-Z0-9]*|[0-9]+)