Frogger Language Definition.

**VARIABLES:** Variables are defined at first use and are initialized to 0. The only variable data type is double. Variable identifiers are alphanumeric with underscores starting with an alpha character. So, valid variable names (identifiers) follow the regex [a-zA-Z][a-zA-Z\_]\*, so numerical digits are not allowed.

**BUILT-IN COMMANDS:** The retrieve() command obtains a double value from the user and can be used anywhere that a double is allowed. The display() command prints the argument to the user; display only takes an arithmetic expression or a string literal. The random() command generates a new pseudo random number between 0 and 1. The end command terminates execution.

**OPERATORS:**Arithmetic operators: assignment (=), addition (++), subtraction (--), multiplication (\*\*), division (//), modulus division (%%), integer division (\\), rootation (##), and exponention (^^) listed in order of operations.  
Boolean operators: not (!), less than (<), greater than (>), equal (==), less than or equal (<=), greater than or equal (>=).

**STRINGS:** Strings are only used for display purposes and include only printable characters and the listed escape characters enclosed within single quotes. Escape characters are: &t (tab), &n (new line), &’ (single quote), and && (ampersand). No control characters are allowed.

**KEYWORDS:** retrieve, end, display, random, if, then, and else.

**COMMENTS:** Comments are enclosed within tildes (~) and are completely ignored.

**PROCESSING ORDER:** Each FLOWSTMT is associated with a number (starting at 0 and incremented by 1 until the end of file linearly). Frogger is not a linear language; that is, code is not processed top to bottom. Instead, at the end of each JMPSTMT, control is passed to the FLOWSTMT corresponding to the number obtained by the following process:  
Add up all the printable characters’ ascii codes for the current JMPSTMT (excepting extraneous parens and subsequent spaces within string literals), mod this number by the total number of FLOWSTMTs in the source program. (Note ascii values for comment characters are ignored because comments do not carry over into the CFG. Note also that conditional structures themselves are not included in ascii summation because the JMPSTMTs are structures contained within the conditional structure.)

**CFG**:

1. PROG -> FLOWSTMT FLOWSTMTS
2. FLOWSTMTS -> FLOWSTMT FLOWSTMTS
3. |
4. FLOWSTMT -> IFSTMT
5. | JMPSTMT
6. IFSTMT -> if ( BOOLEXP ) then NESTEDFLOWSTMT else NESTEDFLOWSTMT
7. JMPSTMT -> display ( STRVAL );
8. | display ( DBLVAL );
9. | end ;
10. | id assign DBLVAL ;
11. BOOLEXP -> DBLVAL BOOLOP DBLVAL
12. | DBLVAL not BOOLOP DBLVAL
13. NESTEDFLOWSTMT -> IFSTMT
14. |JMPSTMT
15. STRVAL -> string
16. DBLVAL -> DBLVAL ADDOP ADDTERM
17. | ADDTERM
18. ADDTERM -> ADDTERM MULOP MULTERM
19. | MULTERM
20. MULTERM -> MULTERM EXPOP EXPTERM
21. | EXPTERM
22. EXPTERM -> dbl
23. | id
24. | ( DBLVAL )
25. | retrieve ( )
26. | random ( )
27. ADDOP -> add
28. | sub
29. MULOP -> mul
30. | div
31. | mod
32. | idiv
33. EXPOP -> rt
34. | exp
35. BOOLOP -> lt
36. | gt
37. | eq
38. | lte
39. | gte

**OBFUSCATION:**

Frogger offers in-line obfuscation if the programmer should choose to further confuse herself/himself. Including the first line of source code as strictly an even number of tildes(~) followed by a carriage return will trigger the de-obfuscator. Note: 0 is considered an even number so if the first character in the source code is a carriage return, the de-obfuscator will run.

Obfuscator Examples (<\n> denotes the new line character within the source code):   
<\n> ~~<\n> ~~~~<\n> ~~~~~~<\n>  
Non-Obfuscated Examples:  
~<\n> (odd number of tildes) ~a~<\n> (‘a’ is not a tilde character) ~~ <\n> (space is not a tilde)

Obfuscation is as follows:  
Each character within identifiers should be ascii incremented based on the number of identifiers occurring prior in the source code. Similarly for each keyword but based on the number of keywords previously occurring. The de-obfuscator will decrement by these counters. Valid symbols are restricted to alphanumeric and the underscore and are incremented in order of ascii value, so order is 0-9A-Z\_a-z. Then if a variable named x\_Dbl is to be used and 6 identifiers have been used between the start of file and the current location, x\_Dbl should be represented (+7) as 4gKis.