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.

**OPERATORS:**Arithmetic operators: addition (++), subtraction (--), multiplication (\*\*), division (//), and assignment (=).   
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, if, and else.

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

**PROCESSING ORDER:** Each STMT and IFSTMT is associated with a line number (starting at 0 and incremented by 1 until the end of file linearly). Frogger is not a linear language; that is, lines of code are not processed top to bottom. Instead, at the end of each statement, control is passed to the statement corresponding to the number obtained by the following process:  
Add up all the printable characters’ ascii codes for the current STMT (excepting extraneous parens and subsequent spaces within string literals), mod this number by the total number of statements 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 because the STMTs are structures under the conditional structure.)

**CFG**:

1. PROG -> IFSTMT STMTS
2. | STMT STMTS
3. STMTS -> IFSTMT STMTS
4. | STMT STMTS
5. |
6. IFSTMT -> if ( BOOLEXP ) STMT else STMT
7. BOOLEXP -> DBLVAL BOOLOPS DBLVAL
8. STMT -> display ( STRVAL );
9. | display ( DBLVAL );
10. | end ;
11. | id assign DBLVAL ;
12. STRVAL -> string
13. DBLVAL -> DBLVAL ADDOP ADDTERM
14. | ADDTERM
15. ADDTERM -> ADDTERM MULOP MULTERM
16. | MULTERM
17. MULTERM -> dbl
18. | id
19. | ( DBLVAL )
20. | retrieve ( )
21. ADDOP -> add
22. | sub
23. MULOP -> mul
24. | div
25. BOOLOPS -> BOOLOP
26. | not BOOLOP
27. BOOLOP -> lt
28. | gt
29. | eq
30. | lte
31. | 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.