1. The program shall be named rpn.rb and should be runnable from the command line using the command ruby rpn.rb.
2. Tokens shall be numbers, variable names, operators, or one of the keywords QUIT, LET, or PRINT.
   1. Check valid number
   2. Check invalid number
   3. Check valid variable name
   4. Check invalid variable name
   5. Check operator
   6. Check invalid operator
   7. Check keywords
3. A number shall consist of one or more digits. All numbers shall be arbitrary-precision (i.e., there shall be no integer overflow - 999999999999999999999999999 shall be considered a valid number and stored as such).
   1. Check valid digit
   2. Check invalid digit
4. Variable names can be a single letter (A-Z) and are case-insensitive (e.g., a and A refer to the same variable).
   1. Check a letter a – z (property-based)
   2. Check something other than a letter
5. Operators can be +, -, /, or \*, for add, subtract, divide, and multiply, respectively.
   1. Check addition
   2. Check subtraction
   3. Check division
   4. Check multiplication
   5. Check an arbitrary operand
6. The keyword QUIT causes the program to end.
   1. Test quit
7. Any lines or tokens after the QUIT keyword are ignored.
   1. Testing quit with lines ignored
8. The keyword LET is followed by a single-letter variable, then an RPN expression. The RPN expression is evaluated and the value of the variable is set to it.
   1. Check value of proper input
   2. Check value of improper input
9. The keyword PRINT is followed by an expression, and the interpreter shall print the result of that expression to standard output (stdout).
   1. Print
10. Keywords shall be case-insensitive (e.g. print, PRINT, or pRiNt are interchangeable)
    1. Send in funny letter print
    2. Send in funny letter let
    3. quit
11. Keywords shall only start a line (e.g., you cannot have a line such as "1 2 + PRINT 3")
12. Variables shall be considered initialized once they have been LET. It shall be impossible to declare a variable without initializing it to some value.
13. Referring to a variable which has not previously been LET (i.e. has not been declared) shall result in the program informing the user that the variable is uninitialized and QUIT the program with error code 1. It should inform the user in the following format: "Line n: Variable x is not initialized." where x is the name of the variable and n is the line number of the file the error occurred in.
14. The exception to the previous requirement is that if this occurs in REPL mode, the user shall be informed, but the line will simply be ignored.
15. If the stack for a given line is empty OR does not contain the correct number of elements on the stack for that operator, and an operator is applied, the program shall inform the user and QUIT the program with error code 2. It should inform the user in the following format: "Line n: Operator o applied to empty stack" where o is the operator used and n is the line number the error occurred in.
16. The exception to the previous requirement is that if this occurs in REPL mode, the user shall be informed, but the line will simply be ignored.
17. If the stack for a given line contains more than one element and the line has been evaluated, the program shall inform the user and QUIT the program with error code 3. It should inform the user in the following format: "Line n: y elements in stack after evaluation" where y is the number of elements in the stack and n is the line number the error occurred in.
18. The exception to the previous requirement is that if this occurs in REPL mode, the user shall be informed, but the line will simply be ignored.
19. If an expression used for initializing a LET variable is invalid, the variable is considered to have not been initialized. For example, "LET A 1 2" is invalid, and A is not initialized.
20. If an invalid keyword is used, the program shall inform the user and QUIT the program with error code 4. It should inform the user in the following format: "Line n: Unknown keyword k" where k is the keyword and n is the line number the error occurred in.
21. The exception to the previous requirement is that if this occurs in REPL mode, the user shall be informed, but the line will simply be ignored.
22. All other errors shall result in the program informing the user of the error and exiting with error code 5. At no point shall the end user of the system see a "raw" exception or stack trace.
23. In REPL mode, the result of each line shall be displayed immediately afterwards before another prompt comes up.
24. In REPL mode, PRINT shall not perform any additional work, as the result of the RPN expression evaluation will already be displayed.
25. In REPL mode, when displaying error messages, the current line shall be considered as the nth command entered. For example, the first line entered shall be considered Line 1, the second line entered Line 2, etc.
26. This program shall minimize real execution time, even at the expense of memory or CPU usage.
27. The "> " string shall be used as the prompt for REPL mode. Note that there is a space after the > character.
28. In case of ambiguity in requirements, the sample output shall be considered the ground truth.
29. When reading multiple files, all of the files shall be concatenated into one large file before processing. For example, when executing file1.rpn and file2.rpn, and file1.rpn initializes a variable A, file2.rpn can use variable A without initializing it. Similarly, if a QUIT is encountered at the end of file1.rpn, the entire program will quit and no lines in file2.rpn shall be executed.
30. Blank lines in files shall be ignored.
31. Lines in files are considered to be 1-indexed, that is, the first line in a file is line number 1, not 0.
32. Variable values shall not be persisted across executions. In other words, if I initialize a variable a, then quit the program and start it again, variable a is no longer initialized.