Programming Assignment #4
COMP 3200
Fall 2023
Prof. Stucki

The provided file calc.py implements a simple calculator using recursive-descent parsing. It supports integers, variables whose names comprise one or more letters, unary negation, and the operators `+`, `-`, `*`, `/`, and `^`. Unassigned variables have the value 0.

Assignment

1. Extend the calculator so that it supports all of the following operators with the correct precedence and associativity.

   Note that the ternary if-then-else expression `x ? y : z` only evaluates *one* of `y` and `z` (never both), and the "or" expression `x | y` and "and" expression `x & y` do not evaluate `y` unless absolutely necessary: these are *short-circuiting* (or *lazy*) operators.

2. Then modify the calc.py program in order that it will read a calculator program (a sequence of expressions, one per line) from a file. It should then loop through the file parsing and evaluating each of the expressions. Once reaching the end of the file your program should then print out the values of all the variables in VARS. Your program should be able to read multiple files in sequence, clearing the VARS dictionary between files.

Expression	Prec	Assoc	Meaning
var = *x*	1	RL	assign *x* to var; result is new value of var
x ? *y* : *z*	2	RL	if *x* is nonzero then *y* else *z*
x | *y*	3	RL	if *x* is zero then *y* else *x* (but only eval *x* once)
x & *y*	4	RL	if *x* is zero then 0 else *y*
! *x*	5	RL	if *x* is zero then 1 else 0
x == *y*	6	LR	if *x* equals *y* then 1 else 0
x != *y*	6	LR	if *x* does not equal *y* then 1 else 0
x < *y*	6	LR	if *x* is less than *y* then 1 else 0
x <= *y*	6	LR	if *x* is less than or equal to *y* then 1 else 0
x > *y*	6	LR	if *x* is greater than *y* then 1 else 0
x >= *y*	6	LR	if *x* is greater than or equal to *y* then 1 else 0
x + *y*	7	LR	*x* plus *y*
x - *y*	7	LR	*x* minus *y*
x * *y*	8	LR	*x* times *y*
x / *y*	8	LR	*x* divided by *y* (integer quotient)
x % *y*	8	LR	remainder when *x* is divided by *y*
- *x*	9	RL	negative *x*
@ *x*	9	RL	absolute value of *x*
x ^ *y*	10	RL	*x* raised to the *y* power
(*x*)	11	—	*x*
var	11	—	current value of var
int	11	—	*int*