The L3 Programming Language

Kevin You

September, 2023

1 About

L3 is an esoteric programming language that originated from the 2023 Carnegie Mellon Informatics and Mathematics Competition's programming contest. This document will contain a specification of the L3 programming language and its extension L3X, and the list of problems from the contest.

2 L3 Language Specifications

2.1 Overview

A L3 program consists of a rectangular grid indexed by (row, column), where each square contains an operation, or is left blank. An operation contains a natural number between 1 and 30 and a direction (up, down, left, right). The program state consists of a variable, which contains a natural number with no prime factors greater than 30, a position on the rectangular grid, and a direction.

2.2 Execution

On a step of the program, the operation of the square occupied by the variable manipulates the variable. Let N be the variable's number, and M the operation's number.

- If the variable's direction is the same as the operation's, then N is multiplied by M, and the variable moves one square in the direction of the operation.
- If the variable's direction is different than the operation's, and N is divisible by M, then N is divided by M, and the variable takes the direction of the operation, and moves one square in this direction.
- If the variable's direction is different than the operation's, and N is *not* divisible by M, then N is unchanged, and the variable takes the direction *opposite* to that of the operation, and moves one square in this direction.

2.3 I/O

Input and output to a L3 program is a natural number with no prime factors greater than 30. Let the dimensions of the grid be height h and width w. The program begins with the variable starting at the top-left corner on square (0,0) moving downwards, and the variable's initial number is the input. The program terminates when the variable moves out of the bottom-right square (h-1,w-1) downwards, and the variable's final number is the output.

2.4 Errors

The program raises an error if the variable moves out of the grid anywhere not at the output or onto a blank square, and the program execution is terminated.

3 Coding in L3

An L3 program is formatted as a comma delimited csv file. Within each cell contains the number followed by the direction as U/D/L/R with no spaces. It is suggested that one uses a spreadsheet tool to write their code, The dimensions of the csv file should match the size of the rectangular grid one intends, so the location of the output matches with what the interpreter expects.

1R	2L	1D

Table 1

1D	1L
1D	3U
1R	2U

Table 2

Shown above are two simple L3 programs. The first example clears the exponent of 2. Given an input $N = 2^x$, its output will be $N = 1$. The second example transfers exponent of 2 to 3. Given an input $N = 2^x$, its output will be 3^x. Both programs are correct, but neither are as small as possible.

4 L3X Language Specifications

4.1 Overview.

The L3 language, while being Turing Complete, is ineffective in handling streams of data, which real-world applications are often built upon. The Extended L3 language, L3X, aims to solve this issue. In L3X, an operation may consists of a special instruction — fork, join, or clear — instead of a number. The program state will now contain multiple variables, and for each join square, a first-in-first-out queues of numbers.

4.2 Execution.

L3X behaves the same as L3 for operations containing a number. Instead, for operations containing a special instruction, L3X does the following:

- On a fork operation, the variable will be duplicated in two, one of the variables will take the direction of the fork operation, the other will take the direction opposite to that of the fork operation, and both variables move one square in their direction.
- On a join operation, if the variable's direction is the same as the operation's, the variable is removed, and its number will be stored in the queue belonging to this square.

- On a join operation, if the variable's direction is different than the operation's, the first number in the queue belonging to this square will be removed and multiplied to the current variable, and the current variable will take the direction of the join operation and move one square.
- On a clear operation, the variable's number will be set to 1, the variable will take the direction of the clear operation and move one square.

The L3X language does not limit the number of variables (unlike L3 previously), but no two variables may occupy the same square simultaneously. Each join operation has its own queue.

4.3 I/O

Input and output to a L3X program is a natural number (the input number and output number) and a stream of natural numbers (the input stream and the output stream).

There are two speical queues, the input queue and the output queue. The input queue must be placed by the programmer at (0, 1) facing downwards. The input stream will begin stored in the input queue. The output queue belongs to a hypothetical join square outside the grid pointing downwards at (h, w-1). Variables moving out of the grid from (h-1,w-1) will be converted to numbers and stored by this join square to become the output stream.

The input and output behave identical to that of L3. A L3X program begins with the input number contained in a variable moving down at (0,0), and terminates when a variable moves down from (h-1,w-1).

4.4 Errors

In addition to erros of L3, an L3X program may raise an error if two variables occupy the same square simultaneously, or if a queue is empty when a number is needed.

4.5 Purity

Unlike L3 programs, which are pure functions, L3X programs may retain numbers in its queues. As standalone programs, purity of L3X programs are not enforced. However, it is important to be aware of this if a L3X program is used as a sub-component of a larger program, where the same block of code is reused.

5 Coding in L3X

In L3X, special instructions fork, join, and clear will be written in the symbols %, &, and ∼, respectively. Otherwise, an L3X program is identical to that of a L3 program.

1R	&D	1R	1D
1D	%L	∼U	1D
1R	1R	1D	1D

Table 3

Shown above is a simple program that takes input $N = 1$, input stream $[2^x]$, outputs $N = 1$ and output stream $[2^x]$. It is worth noting that the number 2^x gets stored in the output queue before the variable containing 1 arrives at the bottom-right. Shown below is a similar but faulty program, which output stream is empty, since the output arrives first.

1R	&D	
1D	%L	∼D
1R	1D	1D

Table 4

6 Problems

The desirability of an L3 program, aside from its correctness, is the size of the code. The area of the rectangle should be as small as possible. The difficulty in L3 to make it an interesting esoteric language and code golf challange is intended to come from routing and optimizing the layout of the code. Your task is to implement the following L30 programs while minimizing the area of code.

Task 1. Implement a L30 program that adds two numbers

Input:	2^x3^y	where x, y are natural numbers
Output:	2^z	where $z = x + y$

Task 2. Implement a L30 program that compares two numbers

Input: 2^x3^y where x, y are natural numbers
Output: 2 if $x > y$, 3 if $x < y$, and 1 if $x = y$

Task 3. Implement a L30 program that multiplies two numbers

Input:	2^x3^y	where x, y are natural numbers
Output:	2^z	where $z = x \times y$

Task 4. Implement a L30 program that performs integer division

Input:	2^x3^y	where x, y are natural numbers and $y > 0$
Output:	2^q3^r	where $x = q \times y + r$ and $0 \leq r < y$

Task 5. Implement a L30 program that computes the greatest common divisor

Input:	2^x3^y	where x, y are natural numbers and $x, y > 0$
Output:	2^z	where $z = \gcd(x, y)$

Task 6. Implement a L30 program that performs square root

Input:	2^x	where x is a natural number
Output:	2^z	where $z = \lfloor\sqrt{x}\rfloor$

7 L30X Problems

Your task is to implement the following L30X programs while minimizing the area of code. For the sake of the contest, purity of the code is not checked.

Task 7. Implement a L30X program that counts down to output stream

Input:	2^n, []	where n is a natural number and $n > 0$
Output:	1, $[2^n, 2^{n-1}, \ldots, 2^1]$	

Task 8. Implement a L30X program that transfers the input stream to the output stream

Input:	2^n, $[2^{x_1}, 2^{x_2}, \ldots, 2^{x_n}]$	where n, x_i are natural numbers
Output:	2^n, $[2^{x_1}, 2^{x_2}, \ldots, 2^{x_n}]$	

Task 9. Implement a L30X program that sums up numbers in the input stream

Input:	2^n, $[2^{x_1}, 2^{x_2}, \ldots, 2^{x_n}]$	where n, x_i are natural numbers
Output:	2^z, []	where $z = x_1 + x_2 + \ldots + x_n$

Task 10. Implement a L30X program that finds the max of numbers in the input stream

Input:	2^n, $[2^{x_1}, 2^{x_2}, \ldots, 2^{x_n}]$	where $n > 0$, x_i are natural numbers
Output:	2^z, []	where $z = \max(x_1, x_2, \ldots, x_n)$

Task 11. Implement a L30X program that finds the median of numbers in the input stream

Input:	2^n, $[2^{x_1}, 2^{x_2}, \ldots, 2^{x_n}]$	where $n > 0$ is odd, x_i are natural numbers
Output:	2^z, []	where $z = \text{median}(x_1, x_2, \ldots, x_n)$