Language Seven Four: By Nia Schimnoski aka Niaschim aka The Schim.

Draft 2

foreward

Language Seven Four, L74, Language74; is designed to be used by hand, to make math easier.
It's also designed to be easily typed on an array of hardware, and hence it has a ridiculous amount of redundancy. It can do a deal of text output and real number output.

it's arbitrarily named, I was gonna call it L77, but then one of its list of properties went up from 7 to 13, and the digital root of 13 is 4.
Then that stopped meaning anything as I kept adding features to the language, so my math language's name is a joke. An in-joke that you know now. It's arbitrary. All names are, but this one especially is.

These are the charts of the features as tables.
I'll also provide examples, clarifications, and a couple equations.

encapsulators

Primary	Primary Alternate	Secondary Alternate	Tertiary Alternate	Description
" "	" "	" "	q _ q	a variable, anything in these quotes is a variable
( )	{ }	en _ en	none	sequential operation you can nest these and the innermost go first
| |	abs abs	val val	none	reframes a variable equation or number as a positive version of the same value.
// \\	//	\\	cc _	a single line or multi line comment
@ @	a a	none	none	when at the start of the line it names that line. Else it refers to a line.

encapsulators go on each side of something to change it's meaning.

input/output

print literal	none	turns everything after this on this line into a message to the console/user, until the next line.
print value	none	takes an array point or a variable and outputs their value to the console/user.
asknum	ask	asks for an integer number input.
write	none	runs once optionally at the end, to write out a list of variables, as their contained values.

talking to and getting talked back to by the code.

Language Seven Four: By Nia Schimnoski aka Niaschim aka The Schim.

Draft 2

digits

Primary	Primary Alternate	Secondary Alternate	Description
	1 one	none	numeric digit equaling 1
	2 two	none	numeric digit equaling 2
	3 three	thr	numeric digit equaling 3
	4 four	for	numeric digit equaling 4
	5 five	fiv	numeric digit equaling 5
	6 six	six	numeric digit equaling 6
	7 seven	sev	numeric digit equaling 7
	8 eight	ght	numeric digit equaling 8
	9 nine	nin	numeric digit equaling 9
	0 zero	zer	numeric digit equaling 0
none	ten	none	numeric digit equaling 10

100 is one hundred, one zero zero, is a syntax error, so use one times ten times ten, instead.

operators

Primary	Primary Alternate	Secondary Alternate	Description
+	plus	none	adds to numbers together
-	minus	none	subtracts the 2nd number from the 1st
÷	divid	/	divides the 1st number by the 2nd
×	times	*	multiplies two numbers
^	power	none	the 1st number to the power of the 2nd
log	lg	oo	the log of one number
sqrt	sq	rt	the square root of one number

math! Divide has divided, multiply has multiplied and times, but I guess I had to invent divid, grammatically. So, there. I did it. It's done.

Language Seven Four: By Nia Schimnoski aka Niaschim aka The Schim.

Draft 2

comparators

Primary	Primary Alternate	Secondary Alternate	Description
!=	ne	none	If not equals
i=	ie	==	if equals
>=	ge	=>	if equal to or greater
<=	le	=<	if equal to or lesser
i>	ig	none	if great
i<	il	none	if lesser
|=	abse	none	If absolute value equals
|<	absl	none	If absolute value less than
|>	absg	none	if absolute value greater than
|<=	absle	none	if absolute value less than or equals
|>=	absge	none	if absolute value greater than or equals
!>	ng	none	if not greater than
!<	nl	none	if not less than
!>=	ngne	none	if not greater than nor equal
!<=	nlne	none	if not less than nor equal

more than you will ever need, baby! So many- we've got redundancies within our redundancies here (that's a good thing!) (now you can code with a broken keyboard)

misc.

the fun stuff. let's you end code pieces, make loops, respond to outcomes. All the good stuff.

Primary	Primary Alternate	Secondary Alternate	Tertiary Alternate	Description
loop[]until	repeat[]until	repeat s s until	loop s s until	loops code in the square bracket until conditions are met.
<arrayName<"ExpansionRate"<"Layers"<"null">>>>	z arrayName z "ExpansionRate" z "Layers" z "null" z z	none	none	Creates a coiled array, of the name arrayName.
<arrayName>["AddressPoint"]	z arrayName z s "AddressPoint" s	none	none	the numeric value of a point in an array (acts like nameless variables)
<arrayName>["AddressPoint1"]iftouch["AddressPoint2"]	Z arrayName z s "AddressPoint1" s iftouch s "Address	none		checks if two points in the same array touch.
if then else	none	none	none	if this, then do that, else do other that. (must include all three)
;	,	.	e	ends the code line.
	_	:	o	does nothing, but sometimes look good.

(might have to zoom in there) (or copy and paste into another document so you can see them either or)

Language Seven Four: By Nia Schimnoski aka Niaschim aka The Schim.

Draft 2

coiled arrays

Do you know what a coiled array is?

It's a term I made up for a thing I independently discovered (it needed a name).

Think of it as a beaded necklace, and then it gets wrapped up in a coil like a snake.

Each bead is a point. Imagine mixing and matching bead sizes and shapes so that the number of beads touching eachother per bead in the coil goes up and down. Beads coiled in a circle that number is six, with beads coiled in a square it's eight.

Imagine that instead of changing the beads, we changed the amount of space in emptiness and our beads were perfect spheres, the math is the same.

Coiled arrays are like that. A tightly coiled line of connected spaces.

Another way to picture them is with gameboards. Imagine a spiral of gameboard spaces, so that moving forwards is the same as traveling the spiral out. Now imagine it so tightly coiled that there is no space between the spaces, and it looks like a big circle; now, someone very far ahead, can go around and be right next to someone further behind. (That actually sounds like a fun game)

That's a coiled array, it's a space where connections can be checked and adjusted like a brain, but simpler obvs. isn't that dope?

Language Seven Four: By Nia Schimnoski aka Niaschim aka The Schim.

Draft 2

important notes

this language is not case sensitive.

Addressing a space in an array that is bigger than the array loops around from the first point until it stops and then it addresses that point.

Changing the size of an array smaller, erases the data that is now outside of the address range, re-enlarging the same array, fills the new addresses with the selected null value (instead of restoring the data).

equations

since you'll be doing this by hand you need a couple equations (literally 2)

//length from array data

```
"ExpansionRate" = 8
"Layers" = 3
Set "x" to 0
Set "length" to 0
Loop [ set "X" to "X" + 1,
        set "length" to "length" + ("x" × "ExpansionRate")] until "x" i= "Layers",

"length" = "length" + 1.

Write "length"
//remember this language is not case sensitive.
```

//The iftouch equation is:

```
"X" = |("point1" ÷ "expansionrate")-("point2" ÷ "expansionrate")|.

If "X" >= 0
Then if "X" <= 1
Then set "1istrue0isfalse" to 1.
Else goto @false@
Else goto @false@
@false@ set "1istrue0isfalse" to 0.

write "1istrue0isfalse" .
//these are little programs to help you utilize the language's more advanced features.
```