59 In Section 2.2 there is a self-describing expression

“““0;0;(0;..29);28;28;(1;..28)”””0;0;(0;..29);28;28;(1;..28)

which evaluates to its own representation.

(a) Write an expression that evaluates to twice its own representation. In other words, it evaluates to its own representation followed by its own representation again.

(b) Make it into a self-printing program. Let's say that !*e* prints the value of expression *e* .

After trying the question, scroll down to the solution.

Before answering the question, here is a guide to the self-reproducing expression

“““0;0;(0;..29);28;28;(1;..28)”””0;0;(0;..29);28;28;(1;..28)

To begin, we have an opening quotation mark “ . It starts a string of characters (a text). The next two opening quotation marks ““ are the way you write one opening quotation mark within a character string, as it says a couple of sentences earlier:

(but a double-quote character within the text must be written twice)

So the first character, character 0, within the string is an opening quotation mark “ . The second character, character 1, is a subscript zero character $_0$. The next character, character 2, is a subscript semi-colon $_;$. And so on. Character 27 is a subscript right parenthesis $_)$. Next we have two closing quotation marks ”” ; that is the way you write one closing quotation mark within a character string, and it is character 28. Then there is one more closing quotation mark ” to end the character string. Let's call that character string s .

s = “““0;0;(0;..29);28;28;(1;..28)”””

After that we have another string of characters; let's call it i .

i = 0;0;(0;..29);28;28;(1;..28)

String i is a subscript, so it is indexing string s .

s_i

The first item in string i is 0 , so that's s_0 , which is an opening quotation mark “ .

The next item in string i is again 0 , so that's s_0 , which is again “ .

Next in string i we have items 0;..29 , so that indexes all of s , from its first character at index 0 , which is “ , to and including its last character at index 28 , which is ” . Remember that 0;..29 includes 0 but not 29 .

Next in string i we have 28 , so that's s_{28} , which is ” .

Next in string i we have another 28 , so that's s_{28} again, which is ” .

And finally in string i we have items 1;..28 , so that indexes all of s except for its first and last characters, which are the opening and closing quotation marks “ and ” .

If you have been keeping track of the characters of s indexed by i , they are:

“““0;0;(0;..29);28;28;(1;..28)”””0;0;(0;..29);28;28;(1;..28)

which is the self-describing expression, also known as a self-reproducing automaton.

(a) Write an expression that evaluates to twice its own representation. In other words, it evaluates to its own representation followed by its own representation again.

§ “““2*(0;0;(0;..33);32;32;(1;..32))”””2*(0;0;(0;..33);32;32;(1;..32))

(b) Make it into a self-printing program. Let's say that $!e$ prints the value of expression e .

§ !“!““0;1;0;1;(1;..34);33;33;(2;..33)””””0;1;0;1;(1;..34);33;33;(2;..33)

Here is a program that prints itself twice with a period between (for sequential composition).

!“!““0;1;0;1;(1;..76);75;76;75;(2;..75);76;0;1;0;1;(1;..76);75;76;75;(2;..75)””.”
0;1;0;1;(1;..76);75;76;75;(2;..75);76;0;1;0;1;(1;..76);75;76;75;(2;..75)

When this program is executed, it prints a program that's twice as long. And when that program is executed, it prints a program that's four times as long as the original. And so on, with exponentially increasing length.