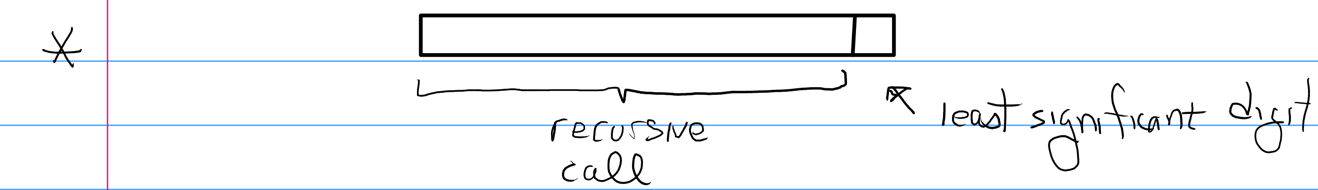
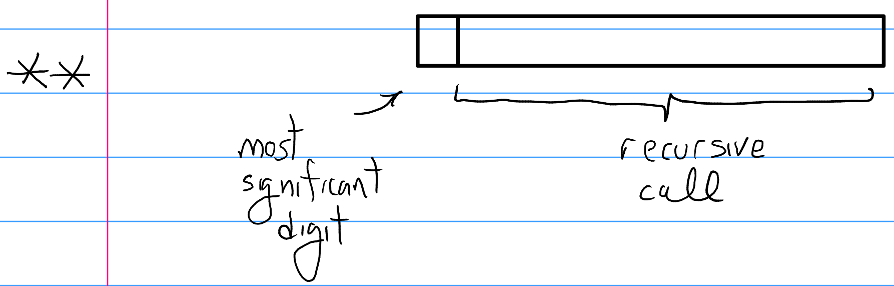
1. ## Reversing the Digits of an Int

1a. ### Recursive Solution

Divide & Conquer ⟶ Perhaps one of these ideas:





Consider * first; we suppose the recursive call reverses all but the rightmost digit, which must then be prepended. Doing this requires knowing how many digits are in the reversed segment — looks like each recursive call will need to return 2 pieces of data — the reversal, and also the number of digits in the reversal.

How would we do this? Two pieces? Could do it with lists, and perhaps other ways, but ... seems like it might be complicated.

We need a workaround. Could we simply start by counting the digits in n? So then $10^{(\#digits(n) - 1)}$ would be the power for (modulo n 10)

So we'd return something like

(+ (* (modulo n 10) (expt 10 (digits-in-(n-1))))
(reverse-digits (quotient n 10)))

$\boxed{\text{doesn't look bad}}$

What about **?

Essentially the same work – but let's sketch it

grab the most signif digit
(again, needs # digits in n)
say (quotient n 10^{same power})

and then reverse the rest

(modulo n 10^{same power})

with result given as

(+ (* (reverse the rest) 10) most-signif. digit)

which do you like best? The take-away is the way in which we avoid returning 2 values from a recursion

Is there a way — without lists — to return multiple values from a function?

Of course — let me sketch 2 of them.

① One idea is to use prime numbers — for example, the pair (p, q) could be returned as the single number

$$2^p 3^q$$

This amounts to an encoding of the (p,q) pair. For example, $(1, 2)$ would be $2^1 3^2 = 2 * 9 = 18$.

[This is the constructor].

For selectors: to get the first element of the pair, just divide by 2 repeatedly until 2 no longer divides the remainder evenly. Amounts to taking the integer base 2 log of the encoded pair.

Similarly, the second element of the pair can be decoded/extracted by repeated division by 3.

This idea extends to triples, and so on: eg

$$2^p 3^q 5^r$$

would encode (p, q, r).

(2) Another way is to use the fact — possibly seen in Discrete Math or Theoretical CS class — that (for example) there are exactly as many pairs of natural numbers as there are natural numbers. This is at first sight surprising, but the idea is simple enough:

One creates a table

spiral which passes through each grid entry exactly once

	0	1	2	3	4	5	...
0	00	01	02	03	04	05	...
1	10	11	12	13	14	15	...
2	20	21	22	23	24	25	...
3	30	31	32	33	34	35	...
4	.	.	.	.	.	.	
5	.	.	.	.	.	.	
⋮	.	.	.	.	.	.	

The encoding of a pair is its position on the spiral. So: (0,0) is 1st, (0,1) is second, ..., (0,3) is 6th etc.

The spiral has a well-known closed form, which makes it possible to decode as well as encode.

Georg Cantor — 1890's

You should all be able to finish the recursive reversal procedure.

What about an iterative design?
Does This help?

nyp	ap
not yet reversed	reversed

N =

can't be right if some of the digits are already in reverse order.

Suggest working from an example (or two)

Suppose N = 12345

and suppose 45 has already been reversed. The question is:

how can we put $\boxed{123}$ and $\boxed{54}$ together to build a GT?

we want to relate the work done already to the total work

$$\boxed{\boxed{54} \cdot (\text{reverse of } 123) = \text{reverse } N}$$

leads to a reasonable first

guess at a GI :

The reversal of N =
[ap] [reversal of nyp]

This turns out to be most/all of the invariant

Termination [nyp] has no digits,
so we'd have
Reversal of N = ap
and we should return ap

[Strong enough ✓]
What about initialization so that the GI
is true? Sure: nyp = N,
ap has no digits.

(no digits?? That's not a number... so a bit more work is needed!)

Preservation:

nxp ← (quotient nxp 10)

ap ← put next digit to the right of ap:

(+ (* ap 10) next-digit)

∘

∘

∘