Quiz 2, Section M Spring 2023

; Problem   Specify, design, develop, code and prove correct an R5RS Scheme program contains? which inputs
; integers m >= 0 and n >= 0, and which outputs #t if all the digits of m occur in n, in the same order,
; subject to multiplicity constraints, and #f otherwise.
; For example,
;; (contains? 54932152 432) = #t: 4, 3, and 2 occur in the right order in 54932152
;; (contains? 5432156 2) = #t
;; (contains? 2435156 52) = #f

what is meant by this?

X not order preserving

order preserving + multiplicity preserving

let's agree that we mean that there exists a 1-1 mapping from the multiset of digits of n to the multiset of digits of m — as drawn with the blue and red arcs.

Consider n = 3 and m = 333

→ (contains? n m) would be true in this case if we eliminated the 1-1 requirement

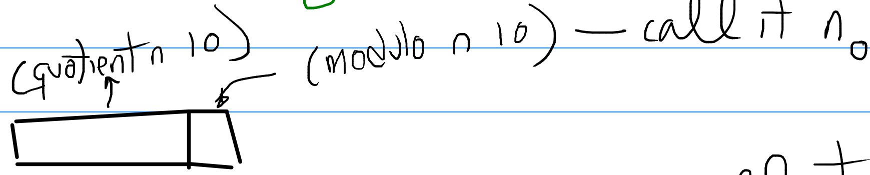
multiplicity

(version 2 will ignore multiplicity)

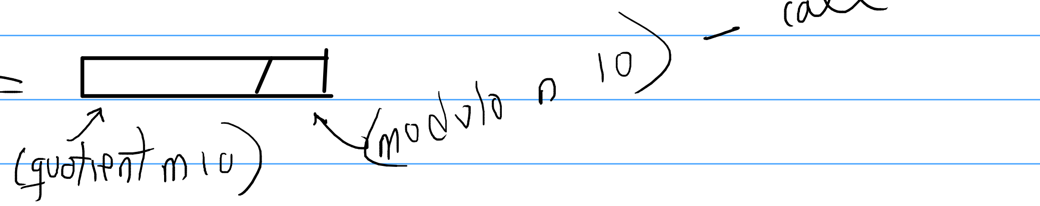
We will solve the problem with the multiplicity requirement first: every digit of m must occur in n at least the same number of times and in the same order.

Divide & Conquer Analysis

We have $n =$



& $m =$



What do we learn from comparing n_0 and m_0?

- What if $n_0 = m_0$?

  If $m < 10$, so that $m = m_0$, we could stop and return #t.

  Otherwise, (contains? n m) is #t iff (contains? (quotient n 10) (quotient m 10))

- What if $n_0 \neq m_0$?

  This means the digits of m can only mate to the digits of (quotient n 10), and the recursive call would be (contains? (quotient n 10) m)

It's looking like this can be structured as an induction on the number of digits in n.

This allows us to figure out the basis case — let's look at $n < 10$ (ie, only 1 digit)

In this case — (contains? n m) = #t iff n = m.
(Again, this uses the multiplicity constraint)

We might be ready to code (?)

```
(define (contains? n m)
  (cond ((< n 10) (= n m))
        (else (let ((n0 (modulo n 10))
                    (m0 (modulo m 10)))

                (cond ((= n0 m0)
                       (if (< m 10)
                           #t
                           (contains? (quotient n 10)
                                      (quotient m 10))))

                      (else
                       (contains? (quotient n 10) m)))
              )
        ))
```

The recursive calls are not guarded, so this is iterative

Cool to observe that we used a divide & conquer development strategy to come up with iterative code!!

So then what is the invariant?

Perhaps something like the following:

(using ghost variables)

$$(\text{contains? } N \ M) = (\text{contains? } n \ m)$$

(Recall the GI for the addition problem discussed in class:

$$A + B = a + b$$)

You should go ahead and check my claim — I am asserting that the development we gave contains the argument showing that this GI is preserved.

However: I haven't forgotten that I said that this program is working by recursion on n.

→ indeed the program can also be verified using a standard data induction on n

(Look at the recursive calls: n is always made smaller)

What's left to do?

→ Is the precondition adequate?

* → Can the problem be solved when multiplicity is ignored?