Vamshi Jandhyala

A Multiple Made of Ones

✠

September 2026

A pigeonhole puzzle from the Monthly Mindbenders.[1]

Problem

Prove that your ten-digit telephone number, if it ends in 1, 3, 7 or 9, has a multiple that, written out in base ten, is nothing but a string of 1s.

Solution

Call the number N, and write R_k for the number made of k ones, so $R_1 = 1$, $R_2 = 11$, $R_3 = 111$ and so on. These are the repunits. The claim is that N divides one of them.

The only thing the last digit tells us is that N is odd and not a multiple of 5, so it shares no factor with 10. That, it turns out, is all the proof needs. The ten digits are decoration: the argument works for every whole number that shares no factor with 10.

Consider the first $N + 1$ repunits, $R_1, R_2, \ldots, R_{N+1}$, and their remainders on division by N. There are only N possible remainders, from 0 to $N - 1$, and there are $N + 1$ repunits, so two of them leave the same remainder. Say R_i and R_j, with $i < j$. Their difference is then a multiple of N, and it has a very particular shape. Subtracting i ones from j ones leaves $j - i$ ones followed by i zeros, so

$$R_j - R_i \;=\; R_{j-i} \times 10^i.$$

For instance $R_5 - R_2 = 11111 - 11 = 11100 = R_3 \times 10^2$. So N divides $R_{j-i} \times 10^i$. But N shares no factor with 10, and so none with 10^i, which means every factor of N must be found in R_{j-i}. Hence N divides R_{j-i}.

$$\boxed{N \text{ divides the repunit } R_{j-i}, \text{ a string of } j - i \text{ ones.}}$$

Why the last digit is exactly the right condition

Every repunit ends in 1, so every repunit is odd and none is a multiple of 5. A number ending in 0, 2, 4, 5, 6 or 8 has a factor of 2 or 5 and can therefore divide no repunit at all. So the puzzle's condition marks exactly the dividing line, both sufficient and necessary.

[1] Peter Winkler, *Monthly Mindbenders*, National Museum of Mathematics, September 2026, momath.org/mindbenders.

A tempting shortcut, and where it breaks

There is a quicker-looking route. Since N shares no factor with 10, some power of ten leaves remainder 1 on division by N, say 10^k. Then N divides $10^k - 1 = 99\ldots9 = 9R_k$, and it is natural to conclude that N divides R_k.

That last step fails whenever N is a multiple of 3. Take $N = 3$: already 10^1 leaves remainder 1, so 3 divides $9R_1 = 9$, but 3 does not divide $R_1 = 1$. The true answer for 3 is $R_3 = 111 = 3 \times 37$. The factor 9 can soak up some or all of the threes in N, and dividing it away loses them.

The shortcut can be repaired by asking instead for a power of ten that leaves remainder 1 on division by $9N$, which also exists. The pigeonhole argument above never needs repairing, because it never divides by anything: it works with the repunits directly.

How long the string is

The same reasoning pins down the shortest string exactly. The repunit R_k equals $(10^k - 1)/9$, so N divides R_k precisely when $9N$ divides $10^k - 1$. The shortest such k is the smallest power for which 10^k leaves remainder 1 on division by $9N$, the multiplicative order of 10 modulo $9N$.

For a ten-digit number this is enormous. Taking (212) 555-0171 as an example, the shortest string of ones that it divides has 8,334,000 digits, and that is on the short side: across 395 random ten-digit numbers ending in 1, 3, 7 or 9 the median length is about 217 million, and only one in seventeen needs fewer than a million ones. The pigeonhole argument promises at most N ones, a few billion, and the true figure is usually well inside that bound while still far too long to write out.

The computation

The check takes 400 numbers ending in 1, 3, 7 or 9. They are the small awkward cases 3, 9, 27, 81 and 37, the example above, and random ten-digit numbers, 143 of them multiples of 3. For each, it finds the multiplicative order k of 10 modulo $9N$ and confirms that 10^k leaves remainder 1 on division by $9N$. That is the same as N dividing R_k. The small cases get a second, slower check that walks the repunits one at a time, updating the remainder as $r \mapsto 10r + 1$, and finds the first repunit divisible by N at exactly the kth.

```
import sympy as sp

def first_repunit(N):
    """walk R_1, R_2, ... modulo N; return the first k with N | R_k"""
    if N % 2 == 0 or N % 5 == 0:
        return None
    r = 0
    for k in range(1, N + 2):
        r = (10 * r + 1) % N
        if r == 0:
            return k

for N in (3, 9, 27, 81, 37):
    assert first_repunit(N) == sp.n_order(10, 9 * N)
```

```
N = 2125550171
k = sp.n_order(10, 9 * N)
print(k, pow(10, k, 9 * N) == 1)   # 8334000 True
print(first_repunit(15), first_repunit(22)) # None None
```

Every one of the 400 numbers has a repunit multiple, including all 143 multiples of 3, and the numbers ending in 5 or an even digit correctly have none. The one pitfall in writing the check is instructive: building R_k itself as a number, with k in the millions, takes far longer than it should, and the powers must be taken modulo $9N$ instead.