First, note that wanting a value to be worth a whole number of pesos is equivalent to wanting its worth, in centavos, to be divisible by 100. We can check this by using the modulo `%` operator.

Consider this brute force strategy. We iterate over all (i, j, k) triples and count how many satisfy `(a[i] + b[j] + c[k]) % 100 == 0`. Unfortunately, there are n^3 such triples, and with n possibly being up to $\approx 3 \times 10^5$, this solution is far too slow.

The key is to note that there are only so many values that "actually matter" for the purposes of this problem. For example, an object worth 1 centavo and an object worth 101 centavos both leave you with 1 centavo as remainder after diving by 100. It's these "remaining centavos" that we're trying to consolidate into a whole number of pesos, at the end of the day, so these two values are "the same."

But since it's 100 pesos to a centavo, there are only 100 possible "remainders" of centavos: 0 to 99.

So, what we can do is "compress" the elements of a into some frequency table `freqA`, where `freqA[r]` counts the number of elements in a which leave a remainder of `r` centavos; again, we reduce each $v \in a$ into `v % 100`. Define `freqB` and `freqC` similarly for the lists b and c.

Then, we only need to iterate over all possible triples of *remainders* (r, s, t) and check how many satisfy `(r + s + t) % 100 == 0`. If such a triple does fit the bill, then by using the rule of product from combinatorics, we know that this contributes `freqA[r] * freqB[s] * freqC[t]` to the total.

At most, there are $100^3 = 10^6$ such triples of remainders, which comfortably passes under the time limit.

```
from collections import defaultdict

n = int(input())

a = [int(x) for x in input().split()]
freqA = defaultdict(int)
for v in a:
    freqA[v % 100] += 1

b = [int(x) for x in input().split()]
freqB = defaultdict(int)
for v in b:
    freqB[v % 100] += 1

c = [int(x) for x in input().split()]
freqC = defaultdict(int)
for v in c:
    freqC[v % 100] += 1

total = 0
for r in freqA:
    for s in freqB:
        for t in freqC:
            if (r + s + t) % 100 == 0:
                total += freqA[r] * freqB[s] * freqC[t]
print(total)
```

As a final gotcha for the C++ and Java users—note that values of size $\approx 3 \times 10^9$ do not fit in a 32-bit integer data type such as `int`. So, make sure that the integers you read from input are stored into 64-bit integers. The answer should be store in a 64-bit integer, too, since it could be up to $\approx (3 \times 10^5)^3$ if all triples are valid.

Exercise: Suppose there are m centavos to a peso (in this problem, $m = 100$). Our current solution runs in $\mathcal{O}(m^3)$. Modify it so that it runs in $\mathcal{O}(m^2)$

Harder Exercise: Suppose there are k lists (in this problem, $k = 3$). Our current solution runs in $\mathcal{O}(nk + m^k)$. Use DP to find a solution that runs in $\mathcal{O}(nk + mk)$.