# Perfect Array

This problem was worth $1000$ points.
The author of this problem is Aryan V S.

**Note:** GitHub does not support LaTex in Markdown. If you want a more readable version of the problem, download the PDF file instead.

## Statement

Aryan and Dhruv are pursuing the Design and Analysis of Algorithms Course at their university. As part of their assignment, the professor assigned them both a problem to solve.

The problem was to implement an efficient algorithm that could compute $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_i \cdot b_j \cdot c_k \ (mod\ 1000000007)$ where $a$, $b$ and $c$ are arrays of size $n$.

They came up with a really efficient solution that only required iterating through the arrays!

They wrote the following solution in C++:

```cpp
int res = 0;
for (int i = 1; i <= n; ++i)
  for (int j = 1; j <= n; ++j)
    for (int k = 1; k <= n; ++k)
      res += a[i] * b[j] * c[k];
res %= 1000000007;
```

Their professor was not very impressed by their solution. Can you manage to impress the professor instead?

## Input Format

The first line contains a single integer $n$ - the size of the arrays.

The second line contains $n$ space separated integers $a_i$.

The third line contains $n$ space separated integers $b_i$.

The fourth line contains $n$ space separated integers $c_i$.

## Output Format

Output the value of `res` as expected from the computation.

## Constraints

$1 \leq n \leq 10^5$

$1 \le a_i,\ b_i,\ c_i \le 10^9$

## Sample Tests

### Sample Test 1

**Input**

```
5
6 4 4 3 10
8 6 4 9 4
8 1 1 5 4
```

**Output**

```
15903
```

## Solution

**Time Complexity:** $O(n)$

**Space Complexity:** $O(n)$

We are required to compute $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_i \cdot b_j \cdot c_k \ (mod\ 1000000007)$.

We can notice that for every $a_i \cdot b_j$, all $c_k$ is being added. As we do when solving mathematical equations, we can "take something common" and write the sum of products as product of sums.

We perform this "taking something common" with $a_i \cdot b_j$ and arrive at $\sum_{i=1}^{n} \sum_{j=1}^{n} a_i \cdot b_j \cdot \sum_{k=1}^{n} c_k \ (mod\ 1000000007)$.

Similarly, we can observe that after the previous step, we can take every $a_i \cdot \sum_{k=1}^{n} c_k$ common in the next step. For the last step, we take every $\sum_{j=1}^{n} b_j \cdot \sum_{k=1}^{n} c_k$ common.

After these steps, we arrive at the equation $\sum_{i=1}^{n} a_i \cdot \sum_{j=1}^{n} b_j \cdot \sum_{k=1}^{n} c_k$ which is the same as the equation given to Aryan and Dhruv by the professor, just written in product of sums form. Computing the output of this equation can be done much more efficiently.

**Solution in C++:**

```cpp
#include <bits/stdc++.h>

void solve () {
  int n;
  std::cin >> n;

  const int64_t mod = 1e9 + 7;
  int64_t sum_A = 0;
  int64_t sum_B = 0;
  int64_t sum_C = 0;

  auto add = [&] (int64_t x, int64_t y) {
    x %= mod;
    y %= mod;
    return (x + y) % mod;
  };

  auto multiply = [&] (int64_t x, int64_t y) {
    x %= mod;
    y %= mod;
    return (x * y) % mod;
  };

  for (int i = 0; i < n; ++i) {
    int64_t x;
    std::cin >> x;
    sum_A = add(sum_A, x);
  }

  for (int i = 0; i < n; ++i) {
    int64_t x;
    std::cin >> x;
    sum_B = add(sum_B, x);
  }

  for (int i = 0; i < n; ++i) {
    int64_t x;
    std::cin >> x;
    sum_C = add(sum_C, x);
  }

  int64_t result = multiply(multiply(sum_A, sum_B), sum_C);
  std::cout << result << '\n';
}

int main () {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  solve();

  return 0;
}
```

**Solution in Python:**

```python
n = int(input())

a = map(int, input().split())
b = map(int, input().split())
c = map(int, input().split())

print(sum(a) * sum(b) * sum(c) % (10 ** 9 + 7))
```