Summing Pieces

HackerRank

Consider an array, A, of length n. We can split A into contiguous segments called *pieces* and store them as another array, B. For example, if $A = [1, 2, 3]$, we have the following arrays of pieces:

- $B = [(1), (2), (3)]$ contains three 1-element pieces.
- $B = [(1, 2), (3)]$ contains two pieces, one having 2 elements and the other having 1 element.
- $B = [(1), (2, 3)]$ contains two pieces, one having 1 element and the other having 2 elements.
- $B = [(1, 2, 3)]$ contains one 3-element piece.

We consider the *value* of a piece in some array B to be $\textit{(sum of all numbers in the piece)} \times \textit{(length of piece)}$, and we consider the *total value* of some array B to be the sum of the values for all pieces in that B. For example, the total value of $B = [(1, 2, 4), (5, 1), (2)]$ is $(1 + 2 + 4) \times 3 + (5 + 1) \times 2 + (2) \times 1 = 35$.

Given A, find the total values for all possible B's, sum them together, and print this sum modulo $(10^9 + 7)$ on a new line.

Input Format

The first line contains a single integer, n, denoting the size of array A.
The second line contains n space-separated integers describing the respective values in A (i.e., $a_0, a_1, \ldots, a_{n-1}$).

Constraints

- $1 \leq n \leq 10^6$
- $1 \leq a_i \leq 10^9$

Output Format

Print a single integer denoting the sum of the total values for all piece arrays (B's) of A, modulo $(10^9 + 7)$.

Sample Input 0

```
3
1 3 6
```

Sample Output 0

```
73
```

Explanation 0
Given $A = [1, 3, 6]$, our piece arrays are:

- $B = [(1), (3), (6)]$, and $\mathit{total\ value} = (1) \times 1 + (3) \times 1 + (6) \times 1 = 10$.
- $B = [(1, 3), (6)]$, and $\mathit{total\ value} = (1 + 3) \times 2 + (6) \times 1 = 14$.
- $B = [(1), (3, 6)]$, and $\mathit{total\ value} = (1) \times 1 + (3 + 6) \times 2 = 19$.
- $B = [(1, 3, 6)]$, and $\mathit{total\ value} = (1 + 3 + 6) \times 3 = 30$.

When we sum all the total values, we get $10 + 14 + 19 + 30 = 73$. Thus, we print the result of $73 \bmod (10^9 + 7) = 73$ on a new line.

Sample Input 1

```
5
4 2 9 10 1
```

Sample Output 1

```
971
```