Again Counting Stars

Translated by Gemini 3

"Said no more counting dollars. We'll be counting stars."

Problem Description

Emperor `__int128` possesses an array $a_1, \dots, a_n$, consisting of **non-negative** integers. There is a cursor initially located at position p, satisfying $1 \le p \le n$.

Emperor `__int128` can perform operations on the array a. For each operation, he may choose to execute one of the following two actions:

- If $p > 1$: set $a_p \leftarrow a_p + 1$, and then move the cursor $p \leftarrow p - 1$.
- If $p < n$: set $a_p \leftarrow a_p - 1$, and then move the cursor $p \leftarrow p + 1$.

Given the array a and the initial cursor position p, Emperor `__int128` can perform an arbitrary number of operations on a. The requirement is that the cursor must eventually return to p, generating a new array b consisting of **non-negative** integers.

For each starting position $1 \le p \le n$, he wants you to calculate the number of distinct valid non-negative arrays b that can be obtained. The answer should be modulo 998244353.

Note: During the operation process, the values a_i **can** be less than 0.

Input Format

The first line of input contains an integer n.

The second line contains n integers $a_1, \dots, a_n$.

Output Format

Output a single line containing n integers. The i-th integer represents the number of possible non-negative arrays b if the initial position was $p = i$, modulo 998244353.

Examples

Example 1

Input

```
3
1 2 1
```

Output

```
5 7 6
```

Explanation

When $p = 1$, the possible distinct non-negative arrays are as follows:

- $[1, 2, 1]$: The cursor makes no moves.
- $[0, 3, 1]$: A possible movement sequence is $1 \to 2 \to 1$.
- $[0, 2, 2]$: A possible movement sequence is $1 \to 2 \to 3 \to 2 \to 1$.
- $[0, 1, 3]$: A possible movement sequence is $1 \to 2 \to 3 \to 2 \to 3 \to 2 \to 1$.
- $[0, 0, 4]$: A possible movement sequence is $1 \to 2 \to 3 \to 2 \to 3 \to 2 \to 3 \to 2 \to 1$.

When $p = 2$, the possible arrays are $[0, 0, 4]$, $[0, 1, 3]$, $[0, 2, 2]$, $[0, 3, 1]$, $[1, 0, 3]$, $[1, 1, 2]$, and $[1, 2, 1]$.

When $p = 3$, the possible arrays are $[0, 0, 4]$, $[0, 1, 3]$, $[0, 2, 2]$, $[1, 0, 3]$, $[1, 1, 2]$, and $[1, 2, 1]$.

Example 2

Input

```
5
2 6 3 0 7
```

Output

```
613 813 820 799 648
```

Example 3

Input

```
7
1 3 4 2 0 1 5
```

Output

```
3060 4915 5161 5166 5118 4847 3862
```

Constraints

This problem uses bundled testing. All logical dependencies between subtasks are enabled.

For all data, it is guaranteed that $1 \le n \le 8000$ and $0 \le a_i \le 5 \cdot 10^8$.

ID	Score	$n \le$	Special Properties
1	8	$\mathbf{7}$	$\sum a_i \le 8$
2	18	$\mathbf{50}$	$\sum a_i \le 3 \cdot 10^5$
3	13	$\mathbf{100}$	$\sum a_i \le 10^7$
4	17	$\mathbf{500}$	$\sum a_i \le 10^7$
5	24	$\mathbf{2500}$	None
6	13	$\mathbf{8000}$	$\sum a_i \le 10^7$
7	7	$\mathbf{8000}$	None

Time Limit: **4** s **Memory Limit:** **1** GiB