E. Balanced

time limit per test: 2 seconds
memory limit per test: 256 megabytes

You are given a **cyclic** array a with n elements, where n is **odd**. In each operation, you can do the following:

- Choose an index $1 \le i \le n$ and increase a_{i-1} by 1, a_i by 2, and a_{i+1} by 1. The element before the first element is the last element because this is a cyclic array.

A cyclic array is called *balanced* if all its elements are equal to each other.

Find any sequence of operations to make this cyclic array balanced or determine that it is impossible. Please note that you **do not** have to minimize the number of operations.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n < 2 \cdot 10^5$, n is **odd**) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case:

- If it is impossible to make the cyclic array balanced, output -1.
- Otherwise, output n integers $v_1, v_2, \ldots, v_n$ ($0 \le v_i \le 10^{18}$) — where v_i denotes the number of operations applied to index i. It can be proved that if any solution exists, then there exists a solution under the given constraints. If there are several solutions under the given constraints, output any of them.

Example

input Copy

```
6
3
2 1 2
3
1 2 3
5
1 2 1 2 1
7
1 2 1 2 1 3 1
9
10000 10000 10000 10000 10000 10001 10002 10001 10000
1
10
```

output Copy

```
0 1 0
2 1 0
2 0 3 0 2
4 2 7 0 8 0 6
```

```
1 1 1 1 1 1 0 1 1
0
```

Note

In the first test case:

- After 1 operation applied to index $i = 2$, the array $a = [3, 3, 3]$.

In the second test case:

- After 2 operations applied to index $i = 1$, the array $a = [5, 4, 5]$.
- After 1 operation applied to index $i = 2$, the array $a = [6, 6, 6]$.

In the third test case:

- After 2 operations applied to index $i = 1$, the array $a = [5, 4, 1, 2, 3]$.
- After 3 operations applied to index $i = 3$, the array $a = [5, 7, 7, 5, 3]$.
- After 2 operations applied to index $i = 5$, the array $a = [7, 7, 7, 7, 7]$.