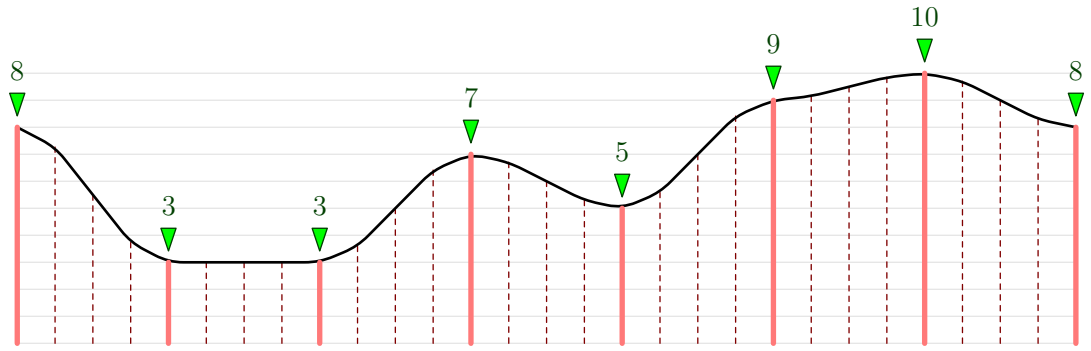
Crazy Roller Coaster (rollercoaster2)

William is again playing his favorite game, *RollerCoaster Typhoon*. He just loves it when family-friendly fun and natural catastrophes mix together.

In the game, there are some predefined *tracks* that can be selected by the player. A track is formed by N junctions, numbered from 1 to N, which are then automatically connected by roller coaster sections.

The i-th junction is located at a specific height H_i. For example, suppose that $H = \{8, 3, 3, 7, 5, 9, 10, 8\}$. For such junction height values, the roller coaster would look something like this:



William loves to scare as much as possible the tiny virtual people going on his roller coaster. He calls his favorite tracks *The Crazy Tracks*. We will say that a track is *crazy* if every consecutive triplet of junctions is **alternating**, that is: for every i such that $2 \leq i \leq N-1$, one of the following is true:

- $H_{i-1} < H_i > H_{i+1}$.
- $H_{i-1} > H_i < H_{i+1}$.

In the example above, there are 3 consecutive triplets of junctions that are **not alternating**:

- $\{8, 3, 3, 7, 5, 9, 10, 8\}$
- $\{8, 3, 3, 7, 5, 9, 10, 8\}$
- $\{8, 3, 3, 7, 5, 9, 10, 8\}$

This means that the track is not *crazy*. Thankfully, the game lets the player change the height of any of the junctions, but this action has a cost: William will have to pay x^2 coins to increase or decrease a junction's height by x units. What is the minimum number of coins that he will have to pay in order to make a given track crazy?

> ☞ Among the attachments of this task you may find a template file `rollercoaster2.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers H_i.

Output

You need to write a single line with an integer: the minimum number of coins to make the track crazy.

Constraints

- $3 \leq N \leq 500$.
- $0 \leq H_i \leq 2000$ for each $i = 1 \ldots N$.
- The minimum number of coins needed is always smaller than 10^9.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (15 points) $H_i \leq 1$ for all i.

– **Subtask 3** (35 points) $N \leq 10$ and $H_i \leq 5$ for all i.

– **Subtask 4** (20 points) $H_i \leq 100$.

– **Subtask 5** (30 points) No additional limitations.

Examples

input	output
8 8 3 3 7 5 9 10 8	23
10 7 6 8 3 7 6 6 2 5 3	1

Explanation

The **first sample case** is the one pictured in the problem statement. It's possible to turn this track *crazy* with 23 coins: decrease 8 ⟶ 5 in first position (9 coins), increase 3 ⟶ 6 in second position (9 coins), decrease 10 ⟶ 8 in second last position (4 coins), increase 8 ⟶ 9 in last position (1 coin).

In the **second sample case**, represented as follows, it's enough to increase the rightmost 6 to 7.