Codeforces Round #666 (Div. 2)

A. Juggling Letters

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n strings $s_1, s_2, \ldots, s_n$ consisting of lowercase Latin letters.

In one operation you can remove a character from a string s_i and insert it to an arbitrary position in a string s_j (j may be equal to i). You may perform this operation any number of times. Is it possible to make all n strings equal?

Input

The first line contains t ($1 \le t \le 10$): the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 1000$): the number of strings.

n lines follow, the i-th line contains s_i ($1 \le |s_i| \le 1000$).

The sum of lengths of all strings in all test cases does not exceed 1000.

Output

If it is possible to make the strings equal, print "YES" (without quotes).

Otherwise, print "NO" (without quotes).

You can output each character in either lowercase or uppercase.

Example

input
4 2 caa cbb 3 cba cba cbb 4 ccab cbac bca acbcc 4 acb caf c cbafc
output
YES NO YES NO

Note

In the first test case, you can do the following:

- Remove the third character of the first string and insert it after the second character of the second string, making the two strings "ca" and "cbab" respectively.
- Remove the second character of the second string and insert it after the second character of the first string, making both strings equal to "cab".

In the second test case, it is impossible to make all n strings equal.

B. Power Sequence

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call a list of positive integers $a_0, a_1, \ldots, a_{n-1}$ a **power sequence** if there is a positive integer c, so that for every $0 \le i \le n-1$ then $a_i = c^i$.

Given a list of n positive integers $a_0, a_1, \ldots, a_{n-1}$, you are allowed to:

- Reorder the list (i.e. pick a permutation p of $\{0, 1, \ldots, n-1\}$ and change a_i to a_{p_i}), then
- Do the following operation any number of times: pick an index i and change a_i to $a_i - 1$ or $a_i + 1$ (i.e. increment or decrement a_i by 1) with a cost of 1.

Find the minimum cost to transform $a_0, a_1, \ldots, a_{n-1}$ into a power sequence.

Input
The first line contains an integer n ($3 \le n \le 10^5$).

The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($1 \le a_i \le 10^9$).

Output
Print the minimum cost to transform $a_0, a_1, \ldots, a_{n-1}$ into a power sequence.

Examples

input
3 1 3 2
output
1

input
3 1000000000 1000000000 1000000000
output
1999982505

Note
In the first example, we first reorder $\{1, 3, 2\}$ into $\{1, 2, 3\}$, then increment a_2 to 4 with cost 1 to get a power sequence $\{1, 2, 4\}$.

C. Multiples of Length

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array a of n integers.

You want to make all elements of a equal to zero by doing the following operation **exactly three** times:

- Select a segment, for each number in this segment we can add a multiple of len to it, where len is the length of this segment (added integers can be different).

It can be proven that it is always possible to make all elements of a equal to zero.

Input
The first line contains one integer n ($1 \le n \le 100\,000$): the number of elements of the array.

The second line contains n elements of an array a separated by spaces: $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output
The output should contain six lines representing three operations.

For each operation, print two lines:

- The first line contains two integers l, r ($1 \le l \le r \le n$): the bounds of the selected segment.
- The second line contains $r - l + 1$ integers $b_l, b_{l+1}, \ldots, b_r$ ($-10^{18} \le b_i \le 10^{18}$): the numbers to add to $a_l, a_{l+1}, \ldots, a_r$, respectively; b_i should be divisible by $r - l + 1$.

Example

input
4 1 3 2 4
output
1 1 -1

```
3 4
4 2
2 4
-3 -6 -6
```

D. Stoned Game

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

T is playing a game with his friend, HL.

There are n piles of stones, the i-th pile initially has a_i stones.

T and HL will take alternating turns, with T going first. In each turn, a player chooses a non-empty pile and then removes a single stone from it. However, one cannot choose a pile that has been chosen in the previous turn (the pile that was chosen by the other player, or if the current turn is the first turn then the player can choose any non-empty pile). The player who cannot choose a pile in his turn loses, and the game ends.

Assuming both players play optimally, given the starting configuration of t games, determine the winner of each game.

Input

The first line of the input contains a single integer t $(1 \leq t \leq 100)$ — the number of games. The description of the games follows. Each description contains two lines:

The first line contains a single integer n $(1 \leq n \leq 100)$ — the number of piles.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 100)$.

Output

For each game, print on a single line the name of the winner, "T" or "HL" (without quotes)

Example

input
2 1 2 2 1 1
output
T HL

Note

In the first game, T removes a single stone from the only pile in his first turn. After that, although the pile still contains 1 stone, HL cannot choose from this pile because it has been chosen by T in the previous turn. Therefore, T is the winner.

E. Monster Invaders

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Ziota found a video game called "Monster Invaders".

Similar to every other shooting RPG game, "Monster Invaders" involves killing monsters and bosses with guns.

For the sake of simplicity, we only consider two different types of monsters and three different types of guns.

Namely, the two types of monsters are:

- a normal monster with 1 hp.
- a boss with 2 hp.

And the three types of guns are:

- Pistol, deals 1 hp in damage to one monster, r_1 reloading time
- Laser gun, deals 1 hp in damage to all the monsters in the current level (including the boss), r_2 reloading time
- AWP, instantly kills any monster, r_3 reloading time

The guns are initially not loaded, and the Ziota can only reload 1 gun at a time.

The levels of the game can be considered as an array $a_1, a_2, \ldots, a_n$, in which **the i-th stage has a_i normal monsters and 1 boss**. Due to the nature of the game, **Ziota cannot use the Pistol (the first type of gun) or AWP (the third type of gun) to**

shoot the boss before killing all of the a_i normal monsters.

If Ziota damages the boss but does not kill it immediately, **he is forced to move out of the current level to an arbitrary adjacent level** (adjacent levels of level i $(1 < i < n)$ are levels $i - 1$ and $i + 1$, the only adjacent level of level 1 is level 2, the only adjacent level of level n is level $n - 1$). Ziota can also choose to move to an adjacent level at any time. **Each move between adjacent levels are managed by portals with d teleportation time.**

In order not to disrupt the space-time continuum within the game, **it is strictly forbidden to reload or shoot monsters during teleportation.**

Ziota starts the game at level 1. The objective of the game is rather simple, to kill all the bosses in all the levels. He is curious about the minimum time to finish the game (assuming it takes no time to shoot the monsters with a loaded gun and Ziota has infinite ammo on all the three guns). Please help him find this value.

Input

The first line of the input contains five integers separated by single spaces: n $(2 \le n \le 10^6)$ — the number of stages, r_1, r_2, r_3 $(1 \le r_1 \le r_2 \le r_3 \le 10^9)$ — the reload time of the three guns respectively, d $(1 \le d \le 10^9)$ — the time of moving between adjacent levels.

The second line of the input contains n integers separated by single spaces $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^6, 1 \le i \le n)$.

Output

Print one integer, the minimum time to finish the game.

Examples

input
4 1 3 4 3 3 2 5 1
output
34

input
4 2 4 4 1 4 5 1 2
output
31

Note

In the first test case, the optimal strategy is:

- Use the pistol to kill three normal monsters and AWP to kill the boss (Total time $1 \cdot 3 + 4 = 7$)
- Move to stage two (Total time $7 + 3 = 10$)
- Use the pistol twice and AWP to kill the boss (Total time $10 + 1 \cdot 2 + 4 = 16$)
- Move to stage three (Total time $16 + 3 = 19$)
- Use the laser gun and forced to move to either stage four or two, here we move to stage four (Total time $19 + 3 + 3 = 25$)
- Use the pistol once, use AWP to kill the boss (Total time $25 + 1 \cdot 1 + 4 = 30$)
- Move back to stage three (Total time $30 + 3 = 33$)
- Kill the boss at stage three with the pistol (Total time $33 + 1 = 34$)

Note that here, we do not finish at level n, but when all the bosses are killed.