



Secuencia de Sigüientes Múltiplos

Ana likes to play with numbers and make sequences with them. Now she is playing making sequences as follows: she first chooses a positive integer m , which will be the first term of the sequence, $a_1 = m$. Then, for all $i \geq 2$, a_i is the smallest multiple of i that is greater than or equal to a_{i-1} . Here are some examples of possible sequences for different values of m :

m	Sequence
1	1,2,3,4,5,6,7, ...
2	2,2,3,4,5,6,7, ...
3	3,4,6,8,10,12, ...
4	4,4,6,8,10,12, ...
5	5,6,6,8,10,12, ...
7	7,8,9,12,15,18, ...
9	9,10,12,12,15,18, ...

Ana has noticed that, for many values of m , there are two consecutive terms of the sequence that are equal, in other words, there exists an i such that $a_i = a_{i+1}$. The case $m = 5$ is an example, since $a_2 = a_3 = 6$. But there are other numbers for which this does not happen. For example, for $m = 1$ or $m = 3$ the sequence has all terms distinct. These values of m for which the resulting sequence does not have two consecutive equal numbers are called *special* numbers.

Can you help Ana to find the n -th special number?

Input and output

The input consists of a positive integer n .

You must print an integer, the n -th special integer.

Examples

Example 1

Input:

1

Output:

1

Example 2

Input:

2

Output:



3

Example 3

Input:

3

Output:

7

Example 4

Input:

123456

Output:

11970633999

Constraints

$1 \leq n \leq 10^7$.

Subtasks

1. (3 points) $n = 5$.
2. (8 points) $n \leq 20$.
3. (9 points) $n \leq 500$.
4. (65 points) $n \leq 10^5$.
5. (15 points) No additional restrictions.



Desordenado

Given a list $a = (a_1, \dots, a_n)$ of n distinct integers, we say that a reordering (or permutation) $b = (b_1, \dots, b_n)$ of the list a is *completely disordered* if every contiguous nonempty sublist of b is not equal to the sublist with the same indices of a but ordered. That is, if for all $1 \leq i \leq j \leq n$, the sequence of numbers $b_i, b_{i+1}, \dots, b_j$ is not equal to the sequence of numbers $a_i, a_{i+1}, \dots, a_j$ but ordered.

For example, given the list $(2, 5, 1, 4, 7)$, the permutation $(5, 7, 4, 2, 1)$ is completely unordered, but the permutations $(7, 1, 4, 5, 2)$ and $(7, 5, 4, 1, 2)$ are not: in the first case, the sublist with $i = 2$ and $j = 4$ $(1, 4, 5)$ coincides with the ordered $(5, 1, 4)$ sublist, in the second case the sublist with $i = 2$ and $j = 2$ (5) coincides with the ordered (5) sublist.

Given a list, you are asked to print a completely unordered permutation.

Input and output

The first line of the input contains the number of cases T .

For each case, the input has one line with an integer n , followed by a second line with n integers $a_1, \dots, a_n$.

For each case, you must print a line with n integers, a completely unordered permutation of a . If there are several possible answers, you can print any of them.

Example

Input:

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

Output:

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

Constraints

$1 \leq T \leq 100$.

$3 \leq n \leq 100$.

$1 \leq a_i \leq 10^9$.

All a_i are distinct.



XXVI Olimpiada Informática Española
Final día 2
desordenado

Subtasks

1. (37 points) $n \leq 8$.
2. (36 points) For all $1 \leq i \leq n$, $a_i = i$.
3. (27 points) No additional restrictions.



Comiendo galletas

Juan has n different kinds of cookies. Of the i -th kind he has a_i cookies. John wants to eat all the cookies, but he must follow a constraint: each day there must be a type of cookie t such that the number of cookies he eats of type t that day must equal the total number of cookies he eats of types other than t that day.

For example, if $n = 3$ and $(a_1, a_2, a_3) = (4, 5, 7)$, a possible way to eat the cookies in two days would be as follows:

1. The first day he eats 4 cookies of the first type, one cookie of the second type, and 3 of the third type. In this day, he has eaten as many cookies of type $t = 1$ as cookies of other different types ($4 = 1 + 3$).
2. The second day he eats 4 cookies of the second type and 4 cookies of the third type. He has eaten as many cookies of type $t = 2$ (or $t = 3$) as cookies of other types he has eaten ($4 = 0 + 4$).

Can Juan eat all of the cookies? If so, you are asked to design a schedule of how many cookies of each type he must eat each day to meet his goal. John prefers to eat the cookies in fewer days, so your schedule should take at most $4n$ days, and to get the maximum score the schedule should take at most $\lceil \frac{n}{2} \rceil$ days.

Input and output

The first line of the input contains the number of cases T .

For each case, the entry has one line with an integer n , followed by a second line with n integers $a_1, \dots, a_n$.

For each case, you must print a line with the word **SI** if it is possible for John to eat all the cookies, or with the word **NO** otherwise. In case the answer is **SI**, you must additionally print a line with an integer d , followed by d lines, each with n integers. The i -th number of line j is the number of cookies of type i that John eats in day j .

Example

Input:

```
4
3
4 5 7
2
2 1
5
1 3 2 11 5
7
1 999999999 1000000000 1000000000 1000000000 1000000000 1000000000
```



Output:

```
SI
2
4 1 3
0 4 4
NO
SI
1
1 3 2 11 5
SI
3
1 999999999 1000000000 0 0 0 0
0 0 0 1000000000 1000000000 0 0
0 0 0 0 1000000000 1000000000
```

Constraints

$1 \leq T \leq 50$.

$2 \leq n \leq 1000$.

$1 \leq a_i \leq 10^9$.

The sum of n for all cases is at most 1000.

The d that you print must satisfy $1 \leq d \leq 4n$.

The number of cookies Juan eats of each type must be a nonnegative integer, less than or equal to the number of cookies remaining of that type in that day.

Subtasks

1. (5 points) $n = 3$.
2. (10 points) $n \leq 6$.
3. (9 points) All a_i are equal.
4. (12 points) For all $1 \leq i \leq n$, $a_i = i$.
5. (14 points) The sum of all a_i is less than or equal to $8n$.
6. (50 points) No additional restrictions.

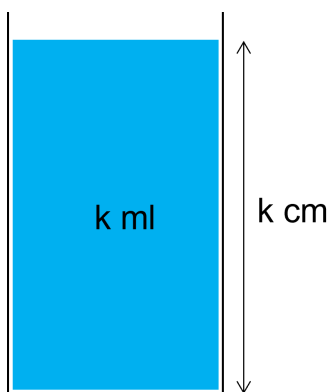
Additionally, the score you get in a subtask depends on the value of d . The score of each subtask is multiplied by a multiplier $M(d, n)$: the value taken from $M(d, n)$ is the smallest among all the cases of the subtask.

$$M(d, n) = \begin{cases} 0 & d > 4n \\ 0.4 + 0.4 \cdot \left(\frac{n}{2d}\right) & 4n \geq d > \left\lceil \frac{n}{2} \right\rceil + 5 \\ 0.8 + 0.04 \cdot \left(\left\lceil \frac{n}{2} \right\rceil + 5 - d\right) & \left\lceil \frac{n}{2} \right\rceil + 5 \geq d > \left\lceil \frac{n}{2} \right\rceil \\ 1.0 & \left\lceil \frac{n}{2} \right\rceil \geq d \end{cases}$$

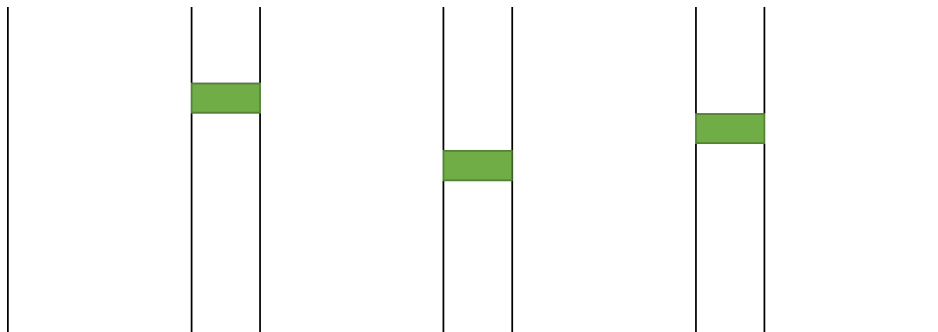


Vasos comunicantes

Alexandra has been learning about volumes in physics this week and wants to run an experiment home. She has n identical plastic cups, which have been designed so that the difference in centimeters between the edge where the water ends in the cup and the bottom of the cup is equal to the number of milliliters of water it contains.

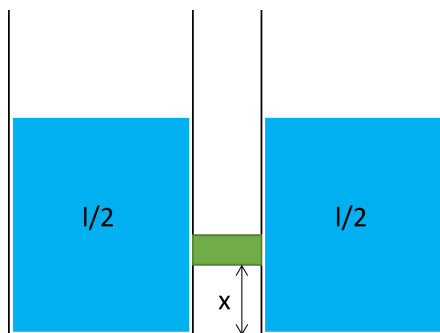


Alexandra has put the glasses in a row and connected the i -th glass with the $i+1$ -th glass for all i , through a horizontal straw. To make the experiment more interesting, she has placed the straws at heights above the bottom of the glass which might not necessarily be equal:



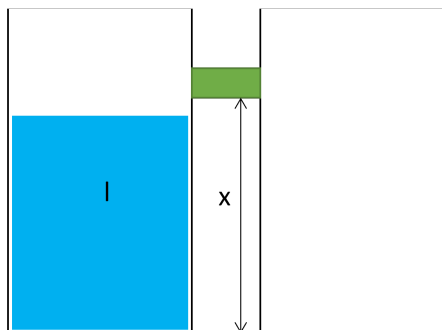


If two contiguous glasses are connected by a straw at a height x , and we pour l milliliters of water into the first glass, if $x \leq \frac{l}{2}$, both glasses will end up with $\frac{l}{2}$ milliliters.

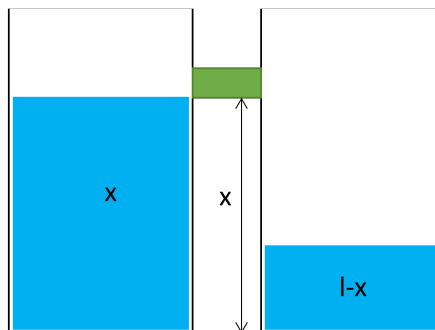


On the other hand, if $x > \frac{l}{2}$, the first glass will end up with $\min(x, l)$ milliliters and the second with l minus this amount.

Caso 1: $\min(x, l) = l$



Caso 2: $\min(x, l) = x$



These patterns will be repeated when there are more than two glasses. That is, when we start pouring water into the first glass, initially all the water remains in the first glass until it reaches the height of the first straw, at which time all the water that continues to be poured fills the second glass, maintaining the height of the first, until the water in the second glass reaches the height of the first or the second straw.

If it reaches the height of the first, from then on the water poured is distributed equally between the first two glasses until the second reaches the height of the second straw, at which time the water that is poured is all filling the third glass until the height of this reaches the height of the second or third straw and the sequence continues.

However, if the water in the second glass reaches the height of the second straw before, the water that is poured will all go to the third glass until it reaches the height of the second or third straw and again we would continue the sequence.

The main two rules to keep in mind is that if we have m isolated glasses and L milliliters of water and the heights of the straws are all equal or inferior to $\frac{L}{m}$ centimeters, there will be $\frac{L}{m}$ milliliters of water in each of the glasses and that, once water begins to fall into the i -th glass, until the height of the water in that glass does not reach that of the previous straw, the height of the water in the previous steady glass will not increase.



We will tell you the list of heights A , where A_i is equal to the height between the i -th glass and the $i + 1$ -th glass (where the first glass is numbered 1). Knowing that she has poured x milliliters of water into the first glass (we assume that no water will ever flow out of the top of the glass, since Alexandra pours it very carefully and there will never be more water than the system can hold), how much water will be in the j -th glass when it stops flowing? You should print this height truncated to be an integer.

Once the list of heights is read, you will receive q queries. However, these queries can come to you in two formats. Before you start reading them, you will receive an integer F , which will be equal to 1 or 2. If $F = 1$, you will receive q queries in the format (x, j) , where x and j represent the explanation above. However, if $F = 2$, you will receive q queries with format $(x + C, j)$, where in the first query $C = 0$ but in the following C will be equal to the result of the previous query. If $F = 2$ you should answer the queries with the original value of x , not with $x + C$.

Input and output

The ticket consists of:

- A line with an integer n .
- A line with the $n - 1$ heights that make up A .
- A line with two integers: q and F .
- q lines with two integers: x y j (or $x + C$ and j if $F = 2$).

You must write one line for each query with the amount of water that will be in the glass.

Examples

Example 1

Input:

```
3
1 2
6 1
1 1
1 2
2 2
3 2
10 2
15 3
```

Output:

```
1
0
1
1
3
5
```



Example 2

Input:

```
10
1 3 4 2 3 4 5 3 1
20 2
0 1
6 3
6 4
1 1
2 2
2 2
4 2
5 3
5 3
20 7
7 3
9 4
15 6
19 8
30 10
100 10
60 10
15 10
3 9
0 5
```

Output:

```
0
0
0
1
0
1
1
0
0
0
1
0
0
0
0
10
5
0
0
0
```



Constraints

- F equals 1 or 2
- $2 \leq n \leq 10^5$
- $1 \leq j \leq n$
- $0 \leq x \leq 10^9$
- $1 \leq q \leq 10^5$
- $1 \leq A_i \leq 10000$ for all i

Subtasks

1. (4 points) $2 \leq n \leq 10$ y $1 \leq q \leq 20$.
2. (7 points) $A_i = A_s$ for all i and all s .
3. (10 points) $A_i \leq A_{i+1}$ for all i .
4. (22 points) $A_i \leq 100$ for all i .
5. (37 points) $F = 1$.
6. (20 points) No additional restrictions.



Reescritura

Félix really likes to play with *strings*. Today he has discovered a way to transform *strings* and asks for your help to find out if he can transform a given *string* into another one.

The possible transformations are given by a set of rules with the following properties:

- The given rules are of the form $c \rightarrow ab$, where a, b, c are characters.
- Each rule $c \rightarrow ab$, can be applied to replace an occurrence of c with ab . For example, if you have the rule $a \rightarrow bc$ we can transform the string `abca` into `bcbca` by applying the rule on the first character.
- These transformations can be applied successively. For example with $a \rightarrow bc$, $b \rightarrow mm$ from the *string* `abca` we can obtain `bcbca`, `mmcbca`, `ammca`, `bcbcbc`...
- We are assured that for each character c there is at most one rule of the form $c \rightarrow ab$.
- We are assured that the given rules *do not form cycles*. That is, for any character c , by applying the rules successively on the string containing only character c , we cannot get to any distinct string containing character c .

Input and output

The first line of the input contains the number of cases T .

For each case there is a first line with n where n is the number of rules, followed by n rows where in the i -th row there is a rule given as `a->bc` where a, b, c are characters. A line follows with a string s and a string t .

For each case a line with “SI” or “NO” (without quotes) must be printed in case s can be converted into t by applying the given rules or not, respectively.

Example

Input:

```
2
1
a->bc
aa
bca
3
a->bc
b->cc
c->mn
abc
acmnx
```

Output:

```
SI
NO
```



Constraints

$$1 \leq T \leq 100$$

$$1 \leq n \leq 59$$

The sum of the lengths of all s and t for all cases is at most $3 \cdot 10^5$.

All characters displayed are lowercase letters (a-z), uppercase letters (A-Z) or numbers (0-9).

Subtasks

1. (17 points) Only **a** and **b** appear as characters.
2. (18 points) $n = 1$.
3. (31 points) Sum of lengths of s and t in each of the cases is at most 60.
4. (34 points) No additional restrictions.