A - AGTC

Source file name: `agtc.py`
Time limit: 2 seconds

Let x and y be two strings over some finite alphabet A. We would like to transform x into y allowing only operations given below:

Deletion: a letter in x is missing in y at a corresponding position.

Insertion: a letter in y is missing in x at a corresponding position.

Change: letters at corresponding positions are distinct.

Certainly, we would like to minimize the number of all possible operations.

The following illustrates a transformation of the string x = `AGTCTGACGC` into y = `AGTAAGTAGGC`:

```
A  G  T  A  A  G  T  *  A  G  G  C
|  |  |           |     |     |  |
A  G  T  *  C  *  T  G  A  C  G  C
```

Deletions are indicated by * in the bottom line, insertions by * in the top line, changes by a blank between the corresponding letters at the top and bottom when they are distinct, and no operation by a | between the corresponding letters at the top and bottom when they are equal. This transformation requires 5 operations, namely, 2 changes, 2 deletions, and 1 insertion.

If we want to minimize the number operations, the transformation:

```
A  G  T  A  A  G  T  A  G  G  C
|  |  |        |     |     |  |
A  G  T  C  T  G  *  A  C  G  C
```

is preferred over the first one because only 4 operations are required: 3 changes and 1 deletion.

By assigning 1 as the cost of any operation performed and 0 if there is no operation performed, write a program that would minimize the number of possible operations to transform an string x into a string y.

Input

Input contains several datasets. Each dataset consists of two lines: the first one containing the string x prefixed with its length, and the second one containing the string y prefixed with its length. The length and the strings in each line are separated by a single space. You can assume that length of y is at least that of x, and that each string has at least one character.

The input must be read from standard input.

Output

For each dataset, output an integer representing the minimum number of operations required to transform x into y.

The output must be written to standard output.

Sample Input	Sample Output
`10 AGTCTGACGC` `11 AGTAAGTAGGC`	`4`

B - The Playboy Chimp

Source file name: `chimp.py`
Time limit: 1 second

Once upon a time, there lived a chimpanzee called Luchu Bandor (a.k.a. *Playboy Chimp*). Luchu was unhappily married to Bunty Mona, a short but cute little lady chimp. Luchu was tall and handsome –he was feeling uncomfortable taking Bunty to public places along with him. People would stare at them all the while.

At one point, Luchu could not stand it anymore and he decided to do some justice to his name. He started looking for a new hope in the Lady Chimps' High School. Every day Luchu would climb up a bamboo tree and wait for the morning drill to start. From there he could see each and every lady chimp doing their routine drill.

Now, Luchu was looking for the tallest lady chimp that would be shorter than him; he would also like to consider someone a little taller than him. But someone of his same height will never be on his list. Every morning Luchu picks up a line of lady chimps and finds the best two according to his set criterion. His job has been made easy by the fact that the lady chimps in each line are ordered by their height, the shortest one is in the front and the tallest one is at the back.

Your task is to help Luchu on one particular day to find two lady chimps: the tallest one shorter than him and the shortest one taller than him.

Input

There will be only one set of input for this problem. The first line of input gives you a number N ($1 \leq N \leq 50000$), the number of lady chimps on the line. In the next line you would have N integers (in the range 1 to 2^{31-1}) giving the heights of the N chimps. There would be a single space after every number. You can assume that the chimps are ordered in non-decreasing order of their heights. In the next line you would have an integer Q ($1 \leq Q \leq 25000$) giving the number of queries. Then in the next line Q queries will follow. Then you would have Q numbers giving the height of Luchu! Don't worry, Luchu is from the land where people can have 3 birthdates; Q heights for a chimpanzee will make no difference here. The Q numbers are listed on a line and their range from 1 to 2^{31-1}, and as before you would find a single space after every query number. The query numbers are not supposed to come in any particular order.

The input must be read from standard input.

Output

For each query height, print two numbers in one line. The first one would be the height of the tallest lady chimp that is shorter than Luchu, and the next number would be the height of the shortest lady chimp that is taller than him. These two numbers are to be separated by a single space. Whenever it is impossible to find any of these two heights, replace that height with an uppercase '*X*'.

The output must be written to standard output.

Sample Input	**Sample Output**
4 1 4 5 7 4 4 6 8 10	1 5 5 7 7 X 7 X

C - Cutting Sticks

Source file name: `cut.py`
Time limit: 1 second

You have to cut a wood stick into pieces. The most affordable company, The Analog Cutting Machinery, Inc. (ACM), charges money according to the length of the stick being cut. Their procedure of work requires that they only make one cut at a time.

It is easy to notice that different selections in the order of cutting can led to different prices. For example, consider a stick of length 10 meters that has to be cut at 2, 4 and 7 meters from one end. There are several choices. One can be cutting first at 2, then at 4, then at 7. This leads to a price of $10 + 8 + 6 = 24$ because the first stick was of 10 meters, the resulting of 8 and the last one of 6. Another choice could be cutting at 4, then at 2, then at 7. This would lead to a price of $10 + 4 + 6 = 20$, which is a better price.

Your boss trusts your computer abilities to find out the minimum cost for cutting a given stick.

Input

The input will consist of several input cases. The first line of each test case will contain a positive number l that represents the length of the stick to be cut. You can assume $l < 1000$. The next line will contain the number n ($n < 50$) of cuts to be made.

The next line consists of n positive numbers c_i ($0 < c_i < l$) representing the places where the cuts have to be done, given in strictly increasing order.

An input case with $l = 0$ will represent the end of the input.

The input must be read from standard input.

Output

You have to print the cost of the optimal solution of the cutting problem, that is the minimum cost of cutting the given stick. Format the output as shown below.

The output must be written to standard output.

Sample Input	**Sample Output**

`100`
`3`
`25 50 75`
`10`
`4`
`4 5 7 8`
`0`
	`The minimum cutting is 200.`
	`The minimum cutting is 22.`

D - Train Swapping

Source file name: `train.py`
Time limit: 3 seconds

At an old railway station, you may still encounter one of the last remaining "train swappers". A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.

Once the carriages are arranged in the optimal order, all the train driver has to do, is drop the carriages off, one by one, at the stations for which the load is meant.

The title "train swapper" stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right.

The first train swapper had discovered that the bridge could be operated with at most two carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite direction, but train carriages can move either way, so who cares).

Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed, is a routine which decides for a given train the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create that routine.

Input

The input contains on the first line the number of test cases (N). Each test case consists of two input lines. The first line of a test case contains an integer L, determining the length of the train ($0 \leq L \leq 25000$). The second line of a test case contains a permutation of the numbers 1 through L, indicating the current order of the carriages. The carriages should be ordered such that carriage 1 comes first, then 2, etc. with carriage L coming last.

The input must be read from standard input.

Output

For each test case output the sentence

```
Optimal train swapping takes S swaps.
```

where S is an integer.

The output must be written to standard output.

Sample Input	**Sample Output**
3 3 1 3 2 4 4 3 2 1 2 2 1	Optimal train swapping takes 1 swaps. Optimal train swapping takes 6 swaps. Optimal train swapping takes 1 swaps.