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 - Luggage

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

Peter and his friends are on holiday, so they have decided to make a trip by car to know the north of Spain. They are seven people and they think that two cars are enough for their luggage.

It's time to leave ... and a heap of suitcases are awaiting out of the cars. The drivers disagree about which suitcase must be put into each boot, because nobody wants one boot to carry more weight than the other one. Is it possible that the two boots load with the same weight? (Obviously without unpacking the suitcases!)

Consider m sets of numbers representing suitcases weights, you must decide for each one, if it is possible to distribute the suitcases into the boots, and the two boots weight the same.

Input

The first line of the input contains an integer, m, indicating the number of test cases. For each test case, there is a line containing n integers ($1 \leq n \leq 100$) separated by single spaces. These integers are the weights of each suitcase. The total sum of the weights of all the suitcases is less or equal to 1000 kilograms.

The input must be read from standard input.

Output

The output consists of m lines. The i-th line corresponds with the i-th set of suitcases weight and contains the string "`YES`" or "`NO`", depending on the possibility that the two boots load with the same weight for the respective test case.

The output must be written to standard output.

Sample Input	**Sample Output**

`3`
`1 2 1 2 1`
`2 3 4 1 2 5 10 50 3 50`
`3 5 2 7 1 7 5 2 8 9 1 25 15 8 3 1 38 45 8 1`
	`NO`
	`YES`
	`YES`

C - Testing the CATCHER

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

A military contractor for the Department of Defense has just completed a series of preliminary tests for a new defensive missile called the CATCHER which is capable of intercepting multiple incoming offensive missiles. The CATCHER is supposed to be a remarkable defensive missile. It can move forward, laterally, and downward at very fast speeds, and it can intercept an offensive missile without being damaged. But it does have one major flaw. Although it can be fired to reach any initial elevation, it has no power to move higher than the last missile that it has intercepted.

The tests which the contractor completed were computer simulations of battlefield and hostile attack conditions. Since they were only preliminary, the simulations tested only the CATCHER's vertical movement capability. In each simulation, the CATCHER was fired at a sequence of offensive missiles which were incoming at fixed time intervals. The only information available to the CATCHER for each incoming missile was its height at the point it could be intercepted and where it appeared in the sequence of missiles. Each incoming missile for a test run is represented in the sequence only once.

The result of each test is reported as the sequence of incoming missiles and the total number of those missiles that are intercepted by the CATCHER in that test.

The General Accounting Office wants to be sure that the simulation test results submitted by the military contractor are attainable, given the constraints of the CATCHER. You must write a program that takes input data representing the pattern of incoming missiles for several different tests and outputs the maximum numbers of missiles that the CATCHER can intercept for those tests. For any incoming missile in a test, the CATCHER is able to intercept it if and only if it satisfies one of these two conditions:

- the incoming missile is the first missile to be intercepted in this test, or
- the missile was fired after the last missile that was intercepted and it is not higher than the last missile which was intercepted.

Input

The input data for any test consists of a sequence of one or more non-negative integers, all of which are less than or equal to 32767, representing the heights of the incoming missiles (the test pattern). The last number in each sequence is −1, which signifies the end of data for that particular test and is not considered to represent a missile height.

The end of data for the entire input is the number −1 as the first value in a test; it is not considered to be a separate test.

The input must be read from standard input.

Output

Output for each test consists of a test number (`Test #1`, `Test #2`, etc.) in a line and the maximum number of incoming missiles that the CATCHER could possibly intercept for the test in a second line, which appears after an identifying message. See the sample output for formatting.

There must be one blank line between output for successive data sets.

The output must be written to standard output.

Sample Input	Sample Output

389
207
155
300
299
170
158
65
-1
23
34
21
-1
-1
	Test #1:
	  maximum possible interceptions: 6
	
	Test #2:
	  maximum possible interceptions: 2

D - The Jackpot

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

As Manuel wants to get rich fast and without too much work, he decided to make a career in gambling. Initially, he plans to study the gains and losses of players, so that, he can identify patterns of consecutive wins and elaborate a win-win strategy. But Manuel, as smart as he thinks he is, does not know how to program computers. So he hired you to write programs that will assist him in elaborating his strategy.

Your first task is to write a program that identifies the maximum possible gain out of a sequence of bets. A bet is an amount of money and is either winning (and this is recorded as a positive value), or losing (and this is recorded as a negative value).

Input

The input set consists of several test cases. The first line in a test case contains a positive number $N \leq 100000$, that gives the length of the sequence. The sencond line in a test case contains N blank-separated integers. Each bet is an integer greater than -1000 and less than 1000.

The input is terminated with $N = 0$.

The input must be read from standard input.

Output

For each given input set, the output will echo a line with the corresponding solution. If the sequence shows no possibility to win money, then the output is the message "`Losing streak.`"

The output must be written to standard output.

Sample Input	**Sample Output**

`5`
`12 -4 -10 4 9`
`3`
`-2 -1 -2`
`0`
	`The maximum winning streak is 13.`
	`Losing streak.`