# A - Exact Sum

*Source file name:* `exact.py`
*Time limit:* 1 second

Peter received money from his parents this week and wants to spend it all buying books. But he does not read a book so fast, because he likes to enjoy every single word while he is reading. In this way, it takes him a week to finish a book.

As Peter receives money every two weeks, he decided to buy two books, then he can read them until receive more money. As he wishes to spend all the money, he should choose two books whose prices summed up are equal to the money that he has. It is a little bit difficult to find these books, so Peter asks your help to find them.

### Input

Each test case starts with $2 \leq N \leq 10000$, the number of available books. Next line will have $N$ integers, representing the price of each book, a book costs less than 1000001. Then there is another line with an integer $M$, representing how much money Peter has. There is a blank line after each test case. The input is terminated by end of input.

*The input must be read from standard input.*

### Output

For each test case you must print the message:

```
Peter should buy books whose prices are i and j.
```

where $i$ and $j$ are the prices of the books whose sum is equal to $M$ and $i \leq j$. You can consider that is always possible to find a solution, if there are multiple solutions print the solution that minimizes the difference between the prices $i$ and $j$. After each test case you must print a blank line.

*The output must be written to standard output.*

| **Sample Input** | **Sample Output** |
|---|---|
| `2`<br>`40 40`<br>`80`<br><br>`5`<br>`10 2 6 8 4`<br>`10` | `Peter should buy books whose prices are 40 and 40.`<br><br>`Peter should buy books whose prices are 4 and 6.` |

# B - 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 $N \leq 10000$, 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`<br>`12 -4 -10 4 9`<br>`3`<br>`-2 -1 -2`<br>`0` | `The maximum winning streak is 13.`<br>`Losing streak.` |

# C - 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<br>3<br>1 3 2<br>4<br>4 3 2 1<br>2<br>2 1 | Optimal train swapping takes 1 swaps.<br>Optimal train swapping takes 6 swaps.<br>Optimal train swapping takes 1 swaps. |