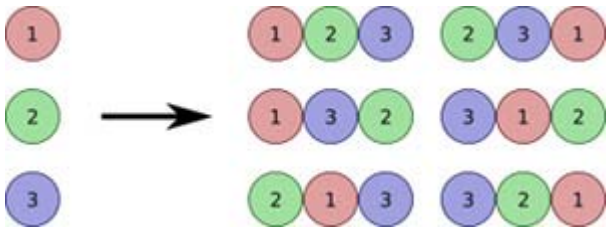
Problem A: Generating Permutations

A permutation on the integers from 1 to *n* is, simply put, a particular rearrangement of these integers. Your task is to generate a given permutation from the initial arrangement 1,2,3,...,*n* using only two simple operations.



- Operation 1: You may swap the first two numbers. For example, this would change the arrangement 3,2,4,5,1 to 2,3,4,5,1.
- Operation 2: You may move the first number to the end of the arrangement. For example, this would change the arrangement 3,2,4,5,1 to 2,4,5,1,3.

Input Format

The input consists of a number of test cases. Each test case begins with a single integer *n* between 1 and 300. On the same line, a permutation of integers 1 through *n* is given where consecutive integers are separated by a single space.

Input is terminated by a line containing '`0`' which should not be processed.

Output Format

For each test case you are to output a string on a single line that describes a sequence of operations. The string itself should consist only of the characters '`1`' and '`2`'. This string should be such that if we start with the initial arrangement 1,2,3,...,*n*-1,*n* and successively apply rules 1 and 2 according to the order they appear in the output, then the resulting permutation is identical to the input permutation.

The output string does not necessarily need to be the shortest such string, but it must be no longer than $2n^2$ characters. If it is possible to generate the permutation using 0 operations, then you may simply output a blank line.

Sample Input

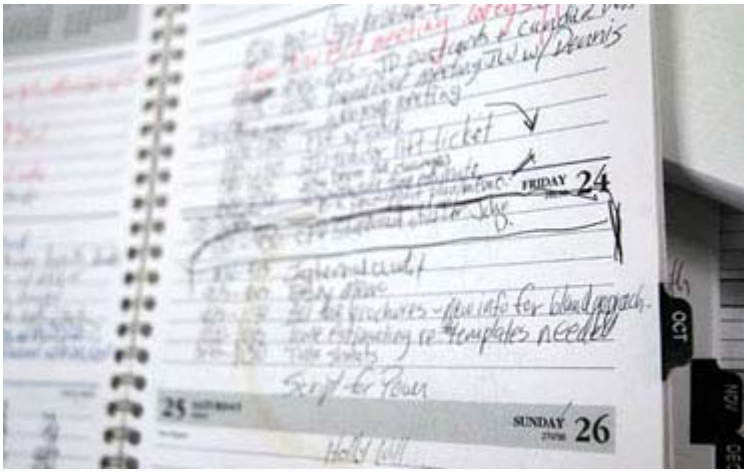
```
3 2 1 3
3 2 3 1
4 4 2 3 1
0
```

Sample Output

```
1
2
12122
```

Zac Friggstad

Problem B: Multitasking



Calendars control our daily lives. For people like me, who are bad at multitasking, it is important to have at most one task planned for any minute of my life. Your job is to make sure that my calendar is free of conflicts for the next one million minutes (just over 99 weeks) of my life. To keep things simple, all times are expressed in minutes from a fixed time 0 representing "now".

In this calendar, there are two types of tasks: one-time tasks and repeating tasks. One-time tasks have a start time and an end time. Repeating tasks have a start time and an end time for their first occurrence, and a repetition interval. Repeating tasks are assumed to keep repeating forever without end. For example, a repeating task with start time 5, end time 8 and repetition interval 100 would be occurring at time intervals [5..8], [105..108], [205..208],…

Tasks are considered to be in conflict if and only if their time intervals overlap, for example [2..5] and [4..6] overlap. "Touching" is OK, for example [2..5] and [5..6] do not overlap.

Input Format

There are approximately 30 test cases. The first line of each test case contains two numbers *n* and *m*. *n* is the number of one-time tasks and *m* the number of repeating tasks. The following *n* lines contain two numbers each, the start and end times respectively of a one-time task. Afterward, *m* more lines similarly describe the repeating tasks by giving their start times, end times, and repetition intervals. Both *n* and *m* are at most 100.

All numbers are integers in the range [0..1000000]. For each task, the end time is guaranteed to be larger than the start time, and the repetition interval is larger than 0.

Input terminates with a line containing '`0 0`' which should not be processed.

Output Format

For each test case, print a single line containing either the words '`NO CONFLICT`' if there are no overlaps between any tasks for minutes 0..1000000, or '`CONFLICT`' if there is at least one overlap.

Sample Input

```
2 0
10 20
20 30
2 0
10 30
20 21
1 1
1000 2000
0 10 1000
```

```
0 0
```

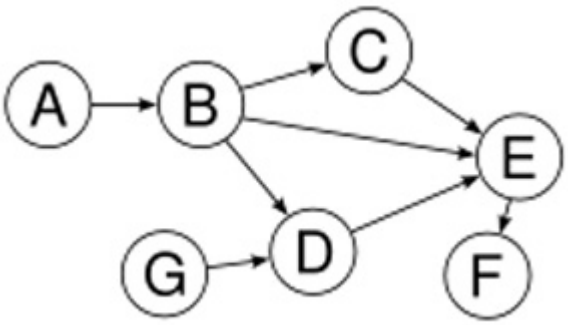
Sample Output

```
NO CONFLICT
CONFLICT
CONFLICT
```

Martin Müller

Problem C: Games Are Important

One of the primary hobbies (and research topics!) among Computing Science students at the University of Alberta is, of course, the playing of games. People here like playing games very much, but the problem is that the games may get solved completely—as happened in the case of Checkers. Generalization of games is the only hope, but worries that they will be solved linger still. Here is an example of a generalization of a two player game which can also be solved.



Suppose we have a directed acyclic graph with some number of stones at each node. Two players take turns moving a stone from any node to one of its neighbours, following a directed edge. The player that cannot move any stone loses the game. Note that multiple stones may occupy the same node at any given time.

Input Format

The input consists of a number of test cases. Each test case begins with a line containing two integers n and m, the number of nodes and the number of edges respectively. ($1 \leq n \leq 1000$, $0 \leq m \leq 10000$). Then, m lines follow, each containing two integers a and b: the starting and ending node of the edge (nodes are labeled from 0 to n-1).

The test case is terminated by n more integers $s_0,\ldots,s_{n-1}$ (one per line), where s_i represents the number of stones that are initially placed on node i ($0 \leq s_i \leq 1000$).

Each test case is followed by a blank line, and input is terminated by a line containing '`0 0`' which should not be processed.

Output Format

For each test case output a single line with either the word '`First`' if the first player will win, or the word '`Second`' if the second player will win (assuming optimal play by both sides).

Sample Input

```
4 3
0 1
1 2
2 3
1
0
0
0

7 7
0 1
0 2
```

```
0 4
2 3
4 5
5 6
4 3
1
0
1
0
1
0
0

0 0
```

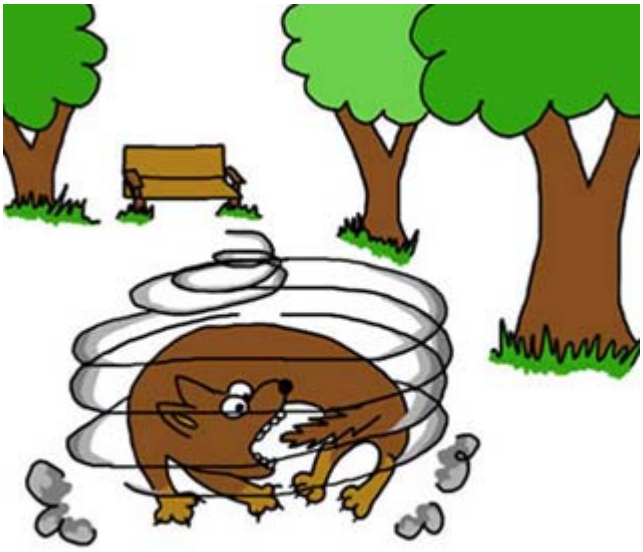
Sample Output

```
First
Second
```

Mohammad Reza Khani

Problem D: The Busy Dog

Rex is a very lucky dog. His owner Stan is very diligent about taking him out for a walk in the park every day. Midway through these walks, Stan takes a break and ties Rex to a pole so he can sit down on a bench for a few minutes.



"Stop bothering me, I'm busy"

Rex is also a very busy dog. When he is tied to this pole, he still likes to run around to look at various curiosities. This often causes his leash to wrap around the pole quite a few times. Once Stan's break is done, he calls Rex back to the bench and then unties his leash. Stan likes to pull the leash tight around the pole, and then see how many times the leash is wrapped around and in which direction.

You will be given the description of the path that Rex follows when he is tied around the pole. For simplicity, this path will be given as a sequence of straight line segments. Remember that Rex always returns to Stan before being untied; this final segment is not explicitly included in the description.

Input Format

The first line of each test case contains a single integer n between 1 and 10,000 describing the number of line segments. The next line consists of two integers x,y which describe the location of the pole. Then n lines follow, where the i'th such line consists of two integers x_i,y_i between -10,000 and 10,000. This means that Rex starts at x_1,y_1 and visits the points in the following manner: after visiting a point x_i, y_i Rex immediately runs in a straight line to point x_{i+1},y_{i+1} for $1 \leq i \leq n$-1. Finally, after visiting the final point x_n,y_n, Rex runs in a straight line to the starting point x_1,y_1.

Input is terminated by a line containing '`0`' which should not be processed.

Output Format

For each test case you are to output a single line consisting of the number of times the leash winds around the pole after Rex's run. Recall that the leash is pulled taught in the direction of Rex's final position after he is done running and you may assume that no knots were formed in this process. If the number of times is $k > 0$, then output $+k$ if the leash is wrapped counter-clockwise around the pole, or output $-k$ if the leash is wrapped clockwise around the pole. If $k = 0$, then simply output 0.

Finally, it may be that Rex actually ran into the pole during his run. If this happens, then you should simply output '`Ouch!`' instead of a number. You can assume the pole is infinitesimally skinny which means that we say Rex runs into the pole if Rex occupies the position x,y at any time during the run.

Sample Input

```
5
0 0
```

```
1 0
-3 1
2 -1
-1 1
-1 -1
4
1 -1
0 0
2 0
2 -2
0 -2
2
0 0
1 1
-1 -1
2
0 0
1 1
1 -1
0
```

Sample Output

```
+2
-1
Ouch!
0
```

Zac Friggstad

Problem E: Ambiguous Forests

The latest assignment in your graph theory course asks you to find the largest collection of edges in a graph which does not contain a cycle. However, you didn't attend the class when the assignment was given so you've asked to see the notes of two of your friends. Unfortunately, one of your friends must have written down the wrong graph because their two graphs are different. In fact, the only thing the two graphs seem to have in common is that they both include the same number of edges m and the edges are numbered from 0 to m-1.

Since you don't know which graph is correct, you have two options. On one hand, you could flip a coin to guess which graph is correct. On the other hand, you could find the largest subset T of the integers from 0 to m-1 so that in each of the two graphs, the edges which are numbered with an integer in T are acyclic. Hoping to guarantee some partial marks, you choose the second option.

Input Format

The first integer k denotes the number of test cases. Each test case begins with three integers n_1, n_2, m where n_1 denotes the number of nodes in the first graph, n_2 denotes the number of nodes in the second graph, and m denotes the number of edges in both graphs.

Then m lines follow where the i'th such line consists of four numbers u_1, v_1, u_2, v_2 where $0 \leq u_j < v_j < n_j$ for each j = 1,2. This means (u_1,v_1) is an edge in the first graph and (u_2,v_2) is an edge in the second graph.

No graph will have more than 50 nodes and the number of edges is at most 200. Furthermore, no graph will have two of the same edge.

Output Format

The output for each test case is a single line beginning with an integer t denoting the size of the largest subset T of integers 0 to m-1 such that neither graph contains a cycle where all edges are numbered with an integer in T. Following this, you should output the numbers of the edges of such a subset in increasing order. Say these integers are $e_1 < e_2 < ... < e_t$. If there are multiple subsets of integers of size t that contain no cycle in either graph, then output the one that minimizes e_t. If there are still multiple such subsets with the same minimum value for e_t, output the one among these that minimizes e_{t-1}, and so on.

Sample Input

```
2
3 4 3
0 1 0 1
1 2 1 2
0 2 2 3
5 5 5
0 1 0 1
1 2 0 2
```

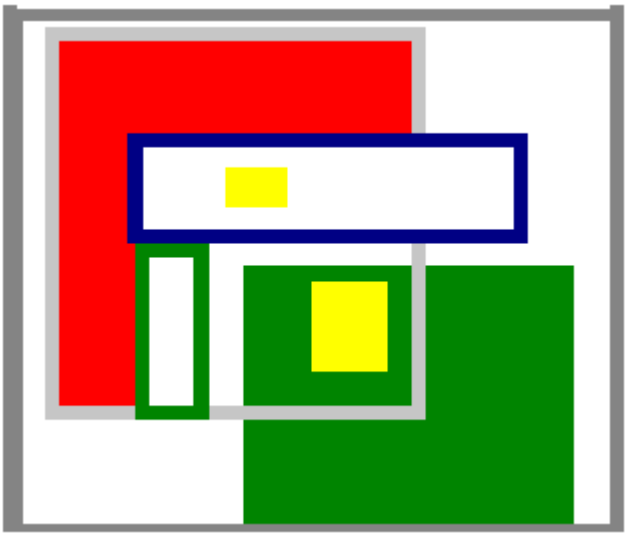
```
1 3 2 4
2 3 3 4
2 4 1 2
```

Sample Output

```
2 0 1
4 0 2 3 4
```

Zac Friggstad

Problem F: Rectangles



Given a set of rectangles in the plane, determine if it is possible to choose one diagonal from each rectangle such that none of the selected diagonals intersect. Two diagonals intersect if they share at least one point. Note that the rectangles themselves are free to intersect.

Input Format

Input consists of several test cases. Each test case begins with an integer n ($1 \leq n \leq 1000$), representing the number of rectangles. This is followed by n lines each describing a rectangle using 8 integer numbers $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$, where each (x_i, y_i) is a vertex. All coordinate values are between -10^9 and 10^9.

The input is terminated by a line containing '0' which should not be processed.

Output Format

For each test case, output a line containing either 'YES' if the selection is possible or 'NO' if not.

Sample Input

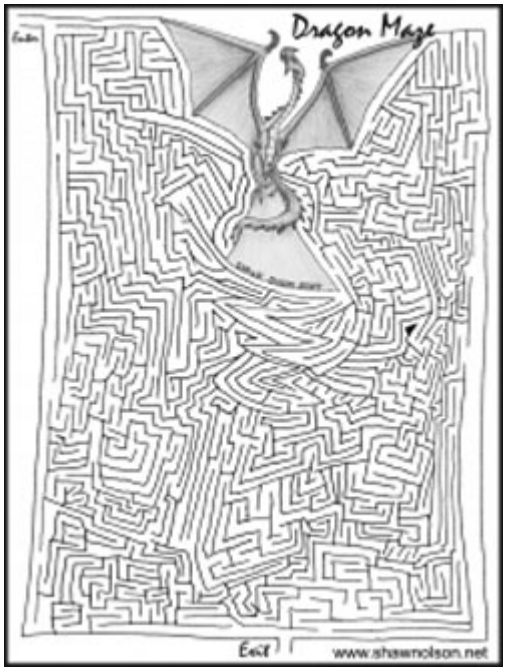
```
4
0    1    1    1    1    0    0    0
1    1    2    1    2    0    1    0
2    3    5    3    5    0    2    0
2    3    3    3    3    2    2    2
7
0    10    10    10    10    0    0    0
10    10    20    10    20    0    10    0
20    30    50    30    50    0    20    0
20    30    30    30    30    20    20    20
30    10    40    10    40    0    30    0
5    0    30    0    30    -10    5    -10
0    0    5    0    5    -10    0    -10
0
```

Sample Output

```
YES
NO
```

Babak Behsaz

Problem G: Maze Escape



Suppose you are stuck in an *n* by *m* maze of cells and you want to escape from it using as few steps as possible. You know that there is a single cell (door cell) that you can escape from. However, the door cell is initially closed and you cannot even move through it. In order to open the door cell, you first have to push a box to a certain position in the maze (button cell).

At each step you can move to one of the north, south, west and east neighboring cells, as long as it is open (not a wall or the closed door cell). In order to push the box, you must be in one of its neighboring cells with an open cell on the other side to move the box into (for example, if you are in the cell north of the box and the cell south of it is open, you can push the box, resulting in both you and the box moving one cell to the south). Pushing the box only counts as one step. While the box is in the button cell, the door cell is open and you can escape the maze through it.

Input Format

Input consists of a number of test cases. Each test case begins with two integers *n* and *m*, the number of rows and columns in the maze respectively ($2 \leq n \leq 20$, $2 \leq m \leq 20$). This is followed by *n* lines each containing exactly *m* characters, representing the maze.

These characters are limited to:

- '`@`', denoting your initial position
- '`d`', denoting the position of the door cell
- '`x`', denoting the initial position of the box
- '`b`', denoting the position of the button cell
- '`#`', denoting a cell containing a wall
- '`.`', denoting an empty cell

Assume that the entire maze is surrounded by walls (in other words, you may never move nor push the box out of the maze) and that there is exactly one each of '`d`', '`b`', '`x`', and '`@`' (this means that the door is initially closed since the box will never start on the button).

Each test case is followed by a blank line, and input is terminated by a line containing '`0 0`' which should not be processed.

Output Format

For each test case, print a line containing the minimum number of steps needed to escape from the maze, or the words '`No way!`' if it is impossible to escape.

Sample Input

```
3 3
```

```
d.b
.@.
x.#

3 5
...d.
.#x#.
...@b

0 0
```

Sample Output

```
No Way!
20
```

Mohammad Reza Khani

Problem H: Net Profit

Thomas has just designed a new game called "Net Profit". The game is played by two players on a "net" of business ventures, each of which offers a certain amount of profit (or loss) in dollars. The term "net" refers to the fact that ventures are connected to each other by randomly generated links (chosen in such a way that all the ventures are connected).

The first player may pick any venture to start, and he scores the associated profit or loss. This venture is now referred to as exhausted. Afterward, the players take turns exhausting ventures and collecting profits, following two simple rules:

1. An exhausted venture may not be selected again (by either player)
2. Only ventures connected to an already exhausted venture are eligible for exhaustion

The game ends once all the ventures are exhausted, and the winner is the player with the greatest profit (or smallest loss). With a given "net" of ventures and associated profits, Thomas would like to know the final outcome of the game assuming optimal play from both players.

Input Format

Input consists of several test cases. Each test case begins with an integer N ($1 \leq N \leq 16$), representing the number of ventures in the net. This is followed by a line containing N integers $p_1, p_2, ..., p_N$; where p_k is the profit associated with venture k ($|p_k| \leq 1000$).

Next is a line containing a non-negative integer M, followed by M lines, each describing a link in the net. Each link description consists of two integers a and b ($1 \leq a, b \leq N, a \neq b$), which means that ventures a and b are linked.

You may assume that the described net connects all the ventures in one component, and that a given link is described at most once (so if link a b is given, link b a will not be).

The input is terminated by a line containing '`0`' which should not be processed.

Output Format

For each test case, output a line with the final result and score of the game assuming optimal play by both players (see the sample output for details).

Sample Input

```
2
25 -20
1
```

```
1 2
3
15 15 -5
2
1 2
2 3
2
30 30
1
1 2
0
```

Sample Output

```
First player wins! 25 to -20.
Second player wins! 15 to 10.
Tie game! 30 all.
```

Sumudu Fernando

Problem I: Splitting Numbers

We define the operation of splitting a binary number *n* into two numbers *a(n),b(n)* as follows. Let $0 \leq i_1 < i_2 < ... < i_k$ be the indices of the bits (with the least significant bit having index 0) in *n* that are 1. Then the indices of the bits of *a(n)* that are 1 are $i_1, i_3, i_5, \ldots$ and the indices of the bits of *b(n)* that are 1 are $i_2, i_4, i_6, \ldots$

For example, if *n* is `110110101` in binary then, again in binary, we have *a* = `010010001` and *b* = `100100100`.

Input Format

Each test case consists of a single integer *n* between 1 and 2^{31}-1 written in standard decimal (base 10) format on a single line. Input is terminated by a line containing '`0`' which should not be processed.

Output Format

The output for each test case consists of a single line, containing the integers *a(n)* and *b(n)* separated by a single space. Both *a(n)* and *b(n)* should be written in decimal format.

Sample Input

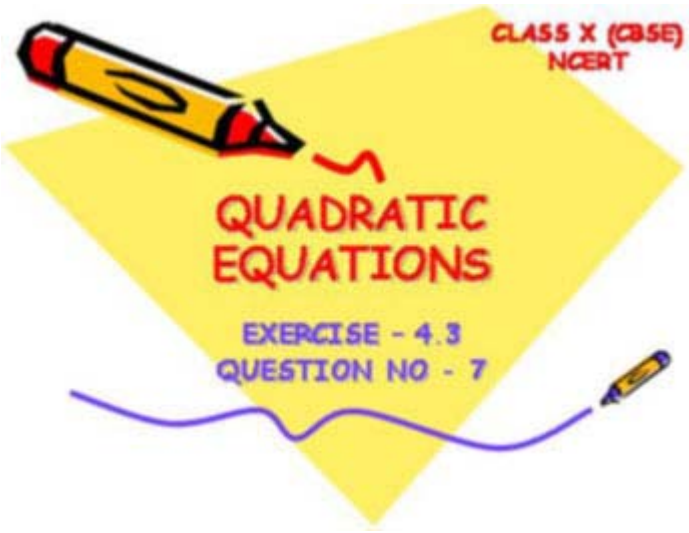
```
6
7
13
0
```

Sample Output

```
2 4
5 2
9 4
```

Zac Friggstad

Problem J: Magic Formula



You are given a quadratic function, $f(n) = a \times n^2 + b \times n + c$. You are also given a divisor d and a limit L. How many of the function values $f(0), f(1),\ldots, f(L)$ are divisible by d?

Input Format

Input consists of a number of test cases. Each test case consists of a single line containing the numbers a b c d L ($-1000 \le a,b,c \le 1000$, $1 < d < 1000000$, $0 \le L < 1000$).

Input is terminated by a line containing '`0 0 0 0 0`' which should not be processed.

Output Format

Print the answer for each test case (the number of function values $f(0), f(1),\ldots, f(L)$ divisible by d) on a separate line.

Sample Input

```
0 0 10 5 100
0 0 10 6 100
1 2 3 4 5
1 2 3 3 5
0 0 0 0 0
```

Sample Output

```
101
0
0
4
```

Martin Müller

Problem K: Through the Desert

Imagine you are an explorer trying to cross a desert. Many dangers and obstacles are waiting on your path. Your life depends on your trusty old jeep having a large enough fuel tank. But how large exactly does it have to be? Compute the smallest volume needed to reach your goal on the other side.

The following events describe your journey:

`Fuel consumption n`
: means that your truck needs *n* litres of gasoline per 100 km. *n* is an integer in the range [1..30]. Fuel consumption may change during your journey, for example when you are driving up or down a mountain.

`Leak`
: means that your truck's fuel tank was punctured by a sharp object. The tank will start leaking gasoline at a rate of 1 litre of fuel per kilometre. Multiple leaks add up and cause the truck to lose fuel at a faster rate.

However, not all events are troublesome in this desert. The following events increase your chances of survival:

`Gas station`
: lets you fill up your tank.

`Mechanic`
: fixes all the leaks in your tank.

And finally:

`Goal`
: means that you have safely reached the end of your journey!

Input Format

The input consists of a series of test cases. Each test case consists of at most 50 events. Each event is described by an integer, the distance (in kilometres measured from the start) where the event happens, followed by the type of event as above.

In each test case, the first event is of the form '`0 Fuel consumption n`', and the last event is of the form '`d Goal`' (*d* is the distance to the goal).

Events are given in sorted order by non-decreasing distance from the start, and the '`Goal`' event is always the last one. There may be multiple events at the same distance—process them in the order given.

Input is terminated by a line containing '`0 Fuel consumption 0`' which should not be processed.

Output Format

For each test case, print a line containing the smallest possible tank volume (in litres, with three digits precision after the decimal point) that will guarantee a successful journey.

Sample Input

```
0 Fuel consumption 10
100 Goal
0 Fuel consumption 5
100 Fuel consumption 30
200 Goal
0 Fuel consumption 20
10 Leak
25 Leak
25 Fuel consumption 30
50 Gas station
70 Mechanic
100 Leak
120 Goal
0 Fuel consumption 0
```

Sample Output

```
10.000
35.000
81.000
```

Martin Müller