Problem A. Circles

Author:	Yury Petrov
Input file:	`circles.in`
Output file:	`circles.out`
Time limit:	3 seconds (*4 seconds for Java*)
Memory limit:	256 Mebibytes

Consider a set of points on the Cartesian plane. For each point in the set, we draw a circle with raduis R. For each circle, find the number of points from the given set which lie on that circle.

Input

The input consists of one or more test cases.

Each test case starts with a line containing two integers: n, the number of circles, and R^2, the square of radius ($1 \le n \le 10^5$, $1 \le R^2 \le 10^9$). The following n lines contain two integers each: the coordinates x_i, y_i of the points ($0 \le x_i, y_i \le 10^9$). It is guaranteed that all points are distinct.

The total number of points in the input will not exceed 10^5. There are at most 1000 test cases in the input. The input will be terminated by a line containing two zeroes which should not be considered as a test case.

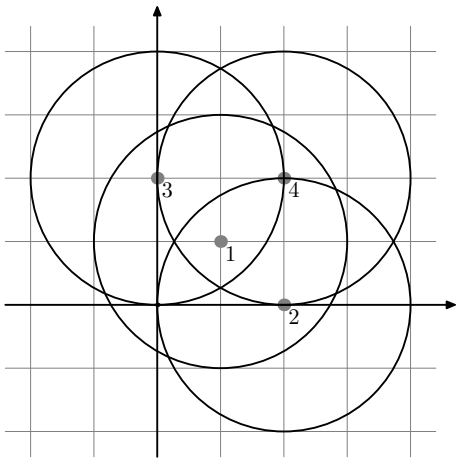
Output

For each test case, write a single line containing n integers: the number of points from the given set lying on each of the n circles in the order their centers are given in the input.

Example

circles.in	circles.out
4 4 1 1 2 0 0 2 2 2 0 0	0 1 1 2

Illustration



Problem B. CRC (Division 1 Only!)

Author:	Anton Maydell
Input file:	`crc.in`
Output file:	`crc.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Research Institute of Given Strings (RIGS) has a large bank of strings. For each string, its cyclic redundancy check code (CRC) is also stored. The hard disks at RIGS are so reliable that each string has at most one corrupted bit. Vasya has to write a program that will repair such errors. To do that, he must solve the following problem.

Consider polynomials in one variable over the field $GF(2)$ (it means that the coefficients can be just 0 and 1, and all calculations for them are performed modulo 2). Let $P(x) = x^{32} + x^{26} + x^{15} + x^7 + 1$. The polynomial $P(x)$ has an amazing property: for any two distinct integers i and j such that $0 \le i, j < 2^{32}$, the polynomials $x^i \bmod P(x)$ and $x^j \bmod P(x)$ are also distinct.

Here, $A(x) \bmod B(x)$ is the remainder of polynomial division of $A(x)$ by $B(x)$. Formally, we have $A(x) \bmod B(x) = R(x)$ where the highest degree of x in $R(x)$ is strictly lower than the highest degree of x in $B(x)$ and there exists a polynomial $Q(x)$ such that $A(x) = Q(x) \times B(x) + R(x)$. As with integer division, there exists exactly one such $Q(x)$ and exactly one $R(x)$ for any polynomial $A(x)$ and any non-zero polynomial $B(x)$. For example, $x^{32} \bmod P(x) = x^{26} + x^{15} + x^7 + 1$ (remember that all coefficients are calculated modulo 2); here, $Q(x) = 1$.

For each given polynomial $S(x)$, Vasya should find the minimal nonnegative k such that $x^k \bmod P(x)$ is equal to $S(x)$. Help him do it.

Input

The input consists of one or more test cases.

Each test case consists of single line containing a polynomial $S(x)$. Each polynomial consists of one or more terms; consecutive terms are separated by character '+'. Each term is written as `x^k` where the power k is an integer such that $0 \le k < 32$. All terms are distinct and given in the order of decreasing k. There are no spaces in the input.

There are at most 200 test cases in the input. The input will be terminated by a line containing a single zero which should not be considered as a test case.

Output

For each test case, write a single line with the answer to the problem.

Example

crc.in	crc.out
x^0	0
x^1	1
x^26+x^15+x^7+x^0	32
x^26+x^21+x^15+x^13+x^7+x^6+x^0	38
x^31+x^25+x^14+x^6	4294967294
0	

Problem C. Integer Points (Division 1 Only!)

Author:	Ivan Kazmenko
Input file:	`integerpoints.in`
Output file:	`integerpoints.out`
Time limit:	3 seconds (*4 seconds for Java*)
Memory limit:	256 Mebibytes

Given three integers p, q and n, find the number of points on the Cartesian plane with integer coordinates satisfying

$$0 \le x \le n \text{ and } 0 \le y \le x \cdot \sqrt{\frac{p}{q}}.$$

Input

On the first line of input there is a single integer t, the number of test cases ($1 \le t \le 100\,000$). After that, t lines follow, each containing one test case. Each test case consists of three integers p, q and n separated by single spaces ($1 \le p, q \le 100$, $0 \le n \le 10^9$).

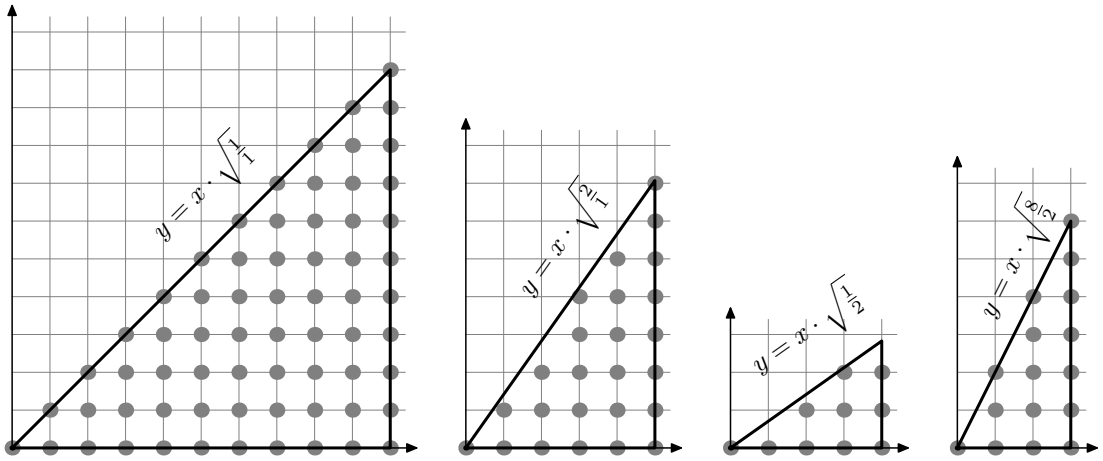
Output

For each test case, your solution should output the number of points with integer coordinates satisfying the constraints on a line by itself.

Example

integerpoints.in	integerpoints.out
4 1 1 10 2 1 5 1 2 4 8 2 3	66 25 10 16

Illustration



Problem D. Game on a Regular Polygon

Author:	Ivan Kazmenko
Input file:	`ngon.in`
Output file:	`ngon.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

A regular polygon is a polygon that is equiangular (all angles are equal in measure) and equilateral (all sides have the same length).

Consider a regular convex polygon with n vertices on a plane. Each vertex is either free or marked. Initially, all vertices are free. Two players make moves in turns. On each move, a player selects k free vertices which are vertices of a regular convex k-gon and marks them. Here, k can be any divisor of n such that $1 < k < n$. There is a special case of $k = 2$: in this game, a regular convex 2-gon is a pair of diametrally opposite vertices of the initial n-gon. There are no regular convex 2-gons if n is odd. A player who cannot make a move loses, the other player is declared a winner.

Given an integer n, find out which player wins if both play optimally.

Input

On the first line of input there is a single integer t, the number of test cases ($1 \le t \le 100$). After that, t lines follow, each containing one test case. Each test case consists of an integer n ($3 \le n \le 10^9$).

Output

For each test case, your solution should output one line containing one word. The word should be either "`First`" if the first player wins, or "`Second`" if the second player wins. Assume that both players play optimally.

Example

ngon.in	ngon.out
2 4 6	Second First

Problem E. Numbers

Author:	Andrey Lopatin
Input file:	`numbers.in`
Output file:	`numbers.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Vasya knows that $2 + 2 \times 2$ is not the same that $(2 + 2) \times 2$. He is given an array A of n integer numbers and his task is very easy: he should say how many different triples (i, j, k) such that $A_i + A_j \times A_k = (A_i + A_j) \times A_k$ he can find. Here, indices i, j and k go through all possible values from 1 to n; note that some of the indices may be equal. Two triples of indices (i_1, j_1, k_1) and (i_2, j_2, k_2) are considered different if at least one of the equalities $i_1 = i_2$, $j_1 = j_2$ and $k_1 = k_2$ is false.

Help Vasya find the number of such triples.

Input

On the first line of input there is a single integer t, the number of test cases ($1 \le t \le 10\,000$). Each test case consists of two lines. The first of these lines contains an integer n ($1 \le n \le 100\,000$), and the second line contains n integers A_i which do not exceed 10^9 by an absolute value. The total length of arrays in all test cases does not exceed 100 000.

Output

For each test case, your solution should output the number of triples described above on a separate line.

Example

numbers.in	numbers.out
2 3 -1 0 1 6 -1 -1 0 0 1 1	15 120

Problem F. Pairs of Numbers

Author:	Yury Petrov
Input file:	`pairs.in`
Output file:	`pairs.out`
Time limit:	5 seconds (*6 seconds for Java*)
Memory limit:	256 Mebibytes

Given an array of integers $a_1, a_2, \ldots, a_n$ and a positive integer X, find the number of pairs of indices (i, j) such that $i < j$ and $\gcd(|a_i - a_j|, X) = 1$.

Here, $\gcd(p, q)$ stands for the greatest common divisor of p and q. Note that $\gcd(0, p) = p$ for any positive integer p.

Input

The input consists of one or more test cases.

Each test case starts with a line containing two integers: n, the number of elements of the array, and X ($1 \le n \le 10^5$, $1 \le X \le 10^8$). The next line contains n integers a_i ($1 \le a_i \le 10^9$).

The total number of elements in all arrays in the input will not exceed 10^5. There are at most 1000 test cases in the input. The input will be terminated by a line containing two zeroes which should not be considered as a test case.

Output

For each test case, write a single line containing the number of pairs.

Example

pairs.in	pairs.out
7 30 1 2 3 4 5 6 7 0 0	6

Problem G. String Packing (Division 1 Only!)

Author:	Anton Maydell
Input file:	`strpack.in`
Output file:	`strpack.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Research Institute of Given Strings (RIGS) asked Vasya to write a string compression program.

Consider a string s over an alphabet of N letters numbered 1, 2, ..., N. For each letter i, Vasya knows the frequency f_i which is how many times letter i occurs in string s. Vasya should assign a non-empty code c_i consisting of zeroes and ones to each letter i. For different i and j, c_i could not be a prefix of c_j. Additionally, length of each code c_i could be no greater than a given integer L. Finally, after replacing each letter i of s by its code c_i, the length of the resulting encoded string should be the minimal possible provided that the restrictions above are met.

Help Vasya to find the lengths of codes c_i.

Input

The input consists of one or more test cases.

The first line of each test case contains two positive integers N and L separated by a single space ($N \le 100$, $L \le 30$). The second line contains N integers f_i separated by single spaces ($1 \le f_i \le 1\,000\,000\,000$).

There are at most 500 test cases in the input. The input will be terminated by a line containing two zeroes which should not be considered as a test case.

Output

For each test case, write two lines. On the first line, output a single integer which is the minimal possible length of the resulting string. On the second line, output N integers: the lengths of codes c_1, c_2, ..., c_N. Separate consecutive integers by a single space.

It is guaranteed that at least one solution exists. In case there are multiple solutions, output any of them.

Example

strpack.in	strpack.out
1 1 1 8 3 1 1 2 3 5 8 13 21 7 7 1 2 3 5 8 13 21 0 0	1 1 162 3 3 3 3 3 3 3 3 124 6 6 5 4 3 2 1

Problem H. Tree Diameter (Division 1 Only!)

Author:	Yury Petrov
Input file:	`treed.in`
Output file:	`treed.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Chainsaw "Druzhba" can be used to cut trees up to 2.5 meters in diameter...

To formulate the problem, let us recall a few definitions from graph theory:

The *distance* between vertices v and u is the minimal number of edges in a path from v to u.

The *eccentricity* of a vertex v is the greatest distance between v and any other vertex.

The *diameter* of a graph is the maximal eccentricity of any vertex in the graph.

Consider an unweighted undirected graph. Initially, the graph consists of a single vertex and does not contain edges. One adds vertices into the graph one after another. Each vertex is added with a single edge connecting it to some other vertex already present in the graph.

Your task is to determine the diameter of the graph after each addition operation.

Input

The input consists of one or more test cases.

Each test case starts with a line containing one integer: n, the number of vertices in the graph after all operations ($2 \le n \le 10^5$). The next line contains $n-1$ integers: $a_2, a_3, \ldots, a_n$ ($1 \le a_i < i$). These numbers define operations: vertex number i will be added with the edge (i, a_i). Vertex number 1 is the initial vertex, other vertices are numbered starting from 2 in the order they are added.

The total number of vertices in all graphs in the input will not exceed 10^5. The input will be terminated by a line containing a single zero which should not be considered as a test case.

Output

For each test case, write a single line containing $n-1$ integers: the diameter of the graph after each operation.

Example

treed.in	treed.out
13 1 1 3 2 1 1 5 7 9 10 4 7 0	1 2 3 4 4 4 5 5 6 7 7 7

Problem I. Universal Path

Author:	Ivan Kazmenko
Input file:	`universalpath.in`
Output file:	`universalpath.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Consider labyrinths of size 10×10 square cells. Each cell is either empty or filled with an impenetrable wall. One of the labyrinth cells is chosen as the starting position, another one as the destination.

The Robot is then placed in the labyrinth at the starting position. The Robot can make steps; a single step moves the Robot to an adjacent cell in one of the four cardinal directions. The Robot cannot step into a wall and cannot leave the labyrinth.

The Robot has a program to execute. The program is just a string consisting of the following commands: make a single step to the North ('`N`'), to the West ('`W`'), to the South ('`S`') or to the East ('`E`'). The commands are executed one after another in the order they appear in the program. For each command, if it is possible to make a step from the Robot's current position in the given direction, she makes that step. Otherwise (if there is a wall in that direction, or the step would make the Robot leave the labyrinth) the Robot just ignores the command.

The Robot's task is to pass the labyrinth, that is, to move to the destination cell. If at some point of time the Robot moves to the destination cell, her task is fulfilled. If the Robot has executed all commands given in the program but has not yet been to the destination cell, the Robot fails to complete the task.

The measure of Robot's accomplishments is the penalty defined as follows. If the task is fulfilled, the penalty is the number of commands executed before the Robot moves to the destination cell for the first time (here, we count both the commands that resulted in making a step and the commands that were ignored). If the Robot fails to pass the labyrinth, the penalty is set to the constant 20 000.

Your task is to write a program for the Robot so that for the certain type of random labyrinths described below, she performs well *on average*, that is, the average penalty does not exceed the given threshold t. Your program should consist of no more than 10 000 commands.

Your program will be checked in the following way. There will be 20 tests. In each test, the input for your solution contains just one integer t. There are also 1000 pre-generated random labyrinths associated with each test, for a total of 20 000 labyrinths; they are however kept secret. After your solution writes a program for the Robot to the output, the checking program executes it on the 1000 pre-generated labyrinths associated with the respective test. If the average penalty for these labyrinths does not exceed the given threshold, your solution passes the test; otherwise, the outcome for the test is "`Wrong Answer`".

The pre-generated labyrinths are generated by the following algorithm. First, 20 distinct cells to mark on the 10×10 grid are picked randomly. After that follow 500 iterations of trying to put a wall. On each iteration, a random cell on the 10×10 grid is picked. If that cell is empty and not marked, a new wall is put on it. After that, if all the marked cells are still reachable from each other, the new wall is kept; otherwise, it is removed to make the cell empty again.

Once the walls are placed, it is time to choose the starting position and the destination. For each pair of cells reachable from each other, the distance is calculated as the length of the shortest path between them; here, the length of a path is the number of steps in that path. A pair of cells with

the longest distance between them is picked randomly. One of the cells of the pair becomes the starting position, and the other becomes the destination.

In the algorithm above, all random picks are independent and have uniform distribution. All labyrinths are generated randomly and independently. Note that your solution does not need to perform well for labyrinths other than generated by this algorithm.

Input

On the first line of input there is a single integer t, the threshold ($1000 \le t \le 1500$). There are exactly 20 tests. It is guaranteed that in test number n, the threshold will be $t = \max(1000, 1550 - n \cdot 50)$. Thus the threshold is 1500 in test 1, 1450 in test 2, 1400 in test 3, ..., 1050 in test 10 and 1000 in tests 11–20. The first 10 tests might show how well your program does. The last 10 tests are to ensure your program does well indeed.

Note that your solution does not need to use the number given in the input, it is there just for convenience.

Output

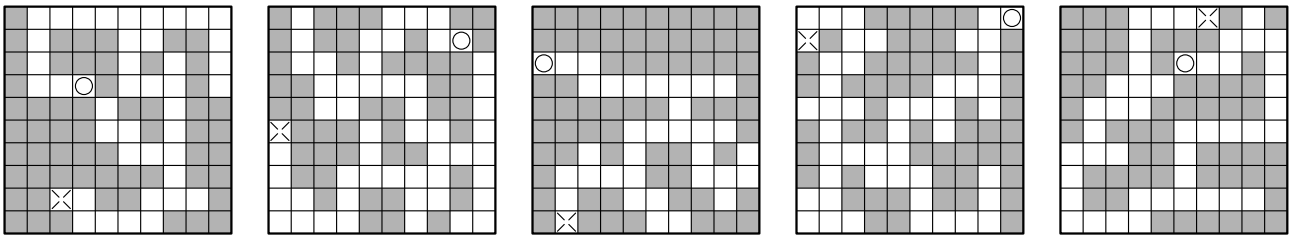
For each test, your solution should output the program for the Robot on a line by itself. Its length should not exceed 10 000 characters. Each character should be either '`N`' for moving to the North, '`W`' for moving to the West, '`S`' for moving to the South, or '`E`' for moving to the East.

In case your solution needs a significant amount of time to perform well, consider precalculating the answer on your local computer and submitting just a program that prints it to the output.

Example

There are no sample tests in this problem.

To illustrate the labyrinth generation algorithm, below are several random labyrinths generated by it.



Problem J. ISBN (Division 2 Only!)

Input file:	`isbn.in`
Output file:	`isbn.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

The International Standard Book Number (ISBN) is a 13-digit code for identifying books. These numbers have a special property for detecting whether the number was written correctly. The *one-three-sum* of a 13-digit number is calculated by multiplying the digits alternately by 1's and 3's and then adding the results. For example, to compute the one-three-sum of the number 8790525616955 we add

$8 \cdot 1 + 7 \cdot 3 + 9 \cdot 1 + 0 \cdot 3 + 5 \cdot 1 + 2 \cdot 3 + 5 \cdot 1 + 6 \cdot 3 + 1 \cdot 1 + 6 \cdot 3 + 9 \cdot 1 + 5 \cdot 3 + 5 \cdot 1$

The special property of an ISBN number is that its one-three-sum is always a multiple of 10. Write a program to check, is 13-digit number correct as ISBN number.

Input

One 13-digit number without leading zeroes.

Output

Print "`Yes`" if number placed in the input file can be used as ISBN number, or "`No`" otherwise.

Examples

isbn.in	isbn.out
8790525616955	Yes
1111111111111	No

Problem K. From Prefix to Postfix (Division 2 Only!)

Input file:	`postfix.in`
Output file:	`postfix.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Prefix notation is a non-conventional notation for writing arithmetic expressions. The standard way of writing arithmetic expressions, also known as infix notation, positions a binary operator between the operands, e. g., `4 + 3`, while in prefix notation the operator is positioned before the operands, e. g., `+ 4 3`. Similarly, the prefix notation for $3 - 2$ is `- 3 2`. A nice property of prefix expressions with binary operators is that parentheses are not required since there is no ambiguity about the order of operations. For example, the prefix representation of $6 - (4 - 1)$ is `- 6 - 4 1`, while the prefix representation of $(6 - 4) - 1$ is `- - 6 4 1`. The prefix notation is also known as Polish notation, due to Jan Lukasiewicz, a Polish logician, who invented it around 1920.

Similarly, in postfix notation, or reverse Polish notation, the operator is positioned after the operands. For example, postfix representation of the infix expression $(6 - 4) - 1$ is `6 4 - 1-`. Your task is to write a program that translates a prefix arithmetic expression into a postfix arithmetic expression.

Input

Each line contains an arithmetic prefix expression. The operators are '`+`' and '`-`', and numbers are all single-digit decimal numbers. The operators and numbers are separated by exactly one space with no leading spaces on the line. The end of input is marked by 0 on a single line. You can assume that each input line contains a valid prefix expression with less than 20 operators and each input file contains no more than 100 examples.

Output

Translate each expression into postfix notation and produce it on a separate line. The numbers and operators are separated by at least one space. The final 0 is not translated.

Example

postfix.in	postfix.out
1 + 1 2 - 2 2 + 2 - 2 1 - - 3 + 2 1 9 0	1 1 2 + 2 2 - 2 2 1 - + 3 2 1 + - 9 -

Problem L. Punchy (Division 2 Only!)

Input file:	`punchy.in`
Output file:	`punchy.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

In the early days of computing, instructions had to be "punched" onto rectangular cards, one instruction per card. This card deck was then fed into a card reader so the program could be read and executed. Students put elastic bands around their card deck, and, often, carried their cards in a box for fear that they would become rearranged, and thus, their program would be incorrect. Poor Bill though... he left his cards right near a window and the wind blew his neat deck of cards all over the place, and thus his program is out of order! Bill decides to pick up the cards in some random order and then execute the program.

Write a program to read and execute the commands in Bill's "new" program.

Input

The programming language that Bill is using has only two variables (A and B) and seven different types of instructions.

Initially, the variables A and B contain the value 0.

There is one instruction per line. An instruction is an integer in the range between 1 and 7, possibly followed by a variable name, which in turn is possibly followed by either a number or a variable.

In all instructions below, the variable X or Y may refer to either A or B. The specific instructions are:

- `1 X n` means set the variable X to the integer value n;
- `2 X` means output the value of variable X to the screen;
- `3 X Y` means calculate $X + Y$ and store the value in variable X;
- `4 X Y` means calculate $X \cdot Y$ and store the value in variable X;
- `5 X Y` means calculate $X - Y$ and store the value in variable X;
- `6 X Y` means calculate the quotient of X/Y and store the value in variable X as an integer, discarding the remainder.
- `7` means stop execution of this program.

You may assume that all division instructions do not cause a division by zero, and that all other operations (including instruction 1) do not cause the computed/stored value to be larger than 10^4 or smaller than -10^4.

(To clarify division of negative numbers, $-3/2$ and $3/-2$ both have quotient -1 and $-3/-2$ has quotient 1.)

Output

Your program should output the value of the indicated variables, one integer per line, until the "stop" instruction has been read in, at which time your program should stop execution.

Example

punchy.in	punchy.out
1 A 3 1 B 4 2 B 2 A 3 A B 2 A 5 A A 2 A 2 B 7	4 3 7 0 4

Problem M. Global Warming (Division 2 Only!)

Input file:	warming.in
Output file:	warming.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 Mebibytes

Your task is to help scientists predict the trend of the global warming. One of the hypotheses they are considering is that over long periods of time, the average temperature follows certain cycles, but each time the cycle starts from a higher temperature level. The temperatures are measured over five-year averages, and expressed in tenths of a degree. For example, if the following five-year averages are observed:

```
3, 4, 6, 4, 5, 7, 5
```

then we can calculate that the temperature changes first 1 tenth up, then 2 up, then 2 down, 1 up, 2 up, and 2 down. There is a cycle of changes of length three which covers all of the temperature differences: $(+1, +2, -2)$. In other words, if we look at the differences starting at the first position, there is a cycle of length three of the form $(+1, +2, -2)$ followed by another cycle of length three of exactly the same form.

By way of another example, suppose the following average temperatures are observed:

```
3, 4, 6, 7.
```

In this case, there is a difference of one up, two up, then one up. We would consider the shortest cycle to be length two in this case: the cycle $(+1, +2)$. Notice that this cycle occurs once, followed by one truncated occurrence of exactly the same cycle.

Your task is to find the shortest such cycle from a given sequence of temperatures.

Input

The input consists of a number of test cases, no more than 5. Each test case starts with the integer n ($1 \le n \le 20$), representing the number of temperatures in a sequence, followed by the sequence of n temperatures. You may assume that each temperature input is an integer in the range between -1000 and 1000 inclusive. The numbers are separated by a single space. The last test case is indicated by a zero and should produce no output.

Output

For each test case, produce the length of the shortest temperature cycle. The cycle always exists, since the whole sequence could be treated as one long cycle.

Example

warming.in	warming.out
7 3 4 6 4 5 7 5 3 1 3 5 3 1 4 5 4 3 4 6 7 0	3 1 2 2