Test Case Tweaking

You are a judge of a programming contest. You are preparing a dataset for a graph problem to seek for the cost of the minimum cost path. You've generated some random cases, but they are not interesting. You want to produce a dataset whose answer is a desired value such as the number representing this year 2010. So you will tweak (which means `adjust') the cost of the minimum cost path to a given value by changing the costs of some edges. The number of changes should be made as few as possible.

A non-negative integer c and a directed graph G are given. Each edge of G is associated with a cost of a non-negative integer. Given a path from one node of G to another, we can define the cost of the path as the sum of the costs of edges constituting the path. Given a pair of nodes in G, we can associate it with a non-negative cost which is the minimum of the costs of paths connecting them.

Given a graph and a pair of nodes in it, you are asked to adjust the costs of edges so that the minimum cost path from one node to the other will be the given target cost c. You can assume that c is smaller than the cost of the minimum cost path between the given nodes in the original graph.

For example, in Figure G.1, the minimum cost of the path from node 1 to node 3 in the given graph is 6. In order to adjust this minimum cost to 2, we can change the cost of the edge from node 1 to node 3 to 2. This direct edge becomes the minimum cost path after the change.

For another example, in Figure G.2, the minimum cost of the path from node 1 to node 12 in the given graph is 4022. In order to adjust this minimum cost to 2010, we can change the cost of the edge from node 6 to node 12 and one of the six edges in the right half of the graph. There are many possibilities of edge modification, but the minimum number of modified edges is 2.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

n m c
f_1 t_1 c_1
f_2 t_2 c_2

f_m t_m c_m

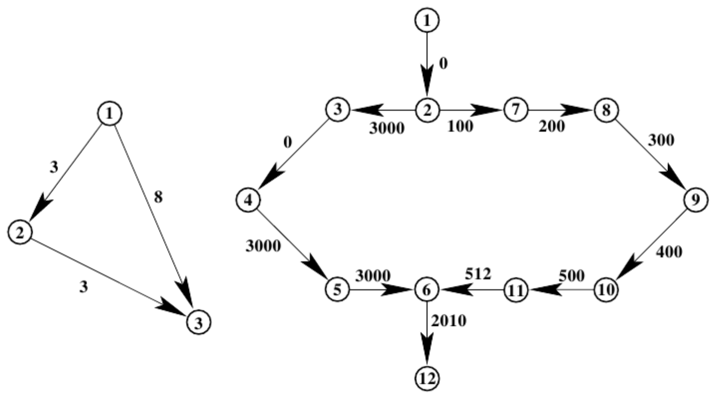


Figure G: Examples 1 and 2 of graphs

The integers n, m and c are the number of the nodes, the number of the edges, and the target cost, respectively, each separated by a single space, where $2 \leq n \leq 100$, $1 \leq m \leq 1000$ and $0 \leq c \leq 100000$.

Each node in the graph is represented by an integer 1 through n.

The following m lines represent edges: the integers f_i, t_i and c_i ($1 \leq i \leq m$) are the originating node, the destination node and the associated cost of the i-th edge, each separated by a single space. They satisfy $1 \leq f_i$, t_i

$\leq n$ and $0 \leq c_i \leq 10000$. You can assume that $f_i \neq t_i$ and $(f_i, t_i) \neq (f_j, t_j)$ when $i \neq j$.

You can assume that, for each dataset, there is at least one path from node 1 to node n, and that the cost of the minimum cost path from node 1 to node n of the given graph is greater than c.

The end of the input is indicated by a line containing three zeros separated by single spaces.

Output

For each dataset, output a line containing the minimum number of edges whose cost(s) should be changed in order to make the cost of the minimum cost path from node 1 to node n equal to the target cost c. Costs of edges cannot be made negative. The output should not contain any other extra characters.

Sample Input

```
3 3 3
1 2 3
2 3 3
1 3 8
12 12 2010
1 2 0
2 3 3000
3 4 0
4 5 3000
5 6 3000
6 12 2010
2 7 100
7 8 200
8 9 300
9 10 400
10 11 500
11 6 512
10 18 1
1 2 9
1 3 2
1 4 6
2 5 0
2 6 10
2 7 2
3 5 10
3 6 3
3 7 10
4 7 6
5 8 10
6 8 2
6 9 11
7 9 3
8 9 9
8 10 8
9 10 1
8 2 1
0 0 0
```

Sample Output

```
1
2
3
```

Where's Wally

Do you know the famous series of children's books named ``Where's Wally"? Each of the books contains a variety of pictures of hundreds of people. Readers are challenged to find a person called Wally in the crowd.

We can consider ``Where's Wally" as a kind of pattern matching of two-dimensional graphical images. Wally's figure is to be looked for in the picture. It would be interesting to write a computer program to solve ``Where's Wally", but this is not an easy task since Wally in the pictures may be slightly different in his appearances. We give up the idea, and make the problem much easier to solve. You are requested to solve an easier version of the graphical pattern matching problem.

An image and a pattern are given. Both are rectangular matrices of bits (in fact, the pattern is always square-shaped). 0 means white, and 1 black. The problem here is to count the number of occurrences of the pattern in the image, i.e. the number of squares in the image exactly matching the pattern. Patterns appearing rotated by any multiples of 90 degrees and/or turned over forming a mirror image should also be taken into account.

Input

The input is a sequence of datasets each in the following format.

w h p
$image\_data$
$pattern\_data$

The first line of a dataset consists of three positive integers w, h and p. w is the width of the image and h is the height of the image. Both are counted in numbers of bits. p is the width and height of the pattern. The pattern is always square-shaped. You may assume $1 \le w \le 1000$, $1 \le h \le 1000$, and $1 \le p \le 100$.

The following h lines give the image. Each line consists of $\lceil w/6 \rceil$ (which is equal to $\lfloor (w+5)/6 \rfloor$) characters, and corresponds to a horizontal line of the image. Each of these characters represents six bits on the image line, from left to right, in a variant of the BASE64 encoding format. The encoding rule is given in the following table. The most significant bit of the value in the table corresponds to the leftmost bit in the image. The last character may also represent a few bits beyond the width of the image; these bits should be ignored.

character	value (six bits)
A-Z	0-25
a-z	26-51
0-9	52-61
+	62
/	63

The last p lines give the pattern. Each line consists of $\lceil p/6 \rceil$ characters, and is encoded in the same way as the image.

A line containing three zeros indicates the end of the input. The total size of the input does not exceed two megabytes.

Output

For each dataset in the input, one line containing the number of matched squares in the image should be output. An output line should not contain extra characters.

Two or more matching squares may be mutually overlapping. In such a case, they are counted separately. On the other hand, a single square is never counted twice or more, even if it matches both the original pattern and its rotation, for example.

Sample Input

```
48 3 3
gAY4I4wA
gIIgIIgg
w4IAYAg4
g
```

```
g
w
153 3 3
kkkkkkkkkkkkkkkkkkkkkkkkkg
SSSSSSSSSSSSSSSSSSSSSSSSSQ
JJJJJJJJJJJJJJJJJJJJJJJJJI
g
Q
I
1 1 2
A
A
A
384 3 2
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/A
CDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/AB
A
A
0 0 0
```

Sample Output

```
8
51
0
98
```

The Two Men of the Japanese Alps

Two experienced climbers are planning a first-ever attempt: they start at two points of the equal altitudes on a mountain range, move back and forth on a single route keeping their altitudes equal, and finally meet with each other at a point on the route. A wise man told them that if a route has no point lower than the start points (of the equal altitudes) there is at least a way to achieve the attempt. This is the reason why the two climbers dare to start planning this fancy attempt.

The two climbers already obtained altimeters (devices that indicate altitude) and communication devices that are needed for keeping their altitudes equal. They also picked up a candidate route for the attempt: the route consists of consequent line segments without branches; the two starting points are at the two ends of the route; there is no point lower than the two starting points of the equal altitudes. An illustration of the route is given in Figure E.1 (this figure corresponds to the first dataset of the sample input).

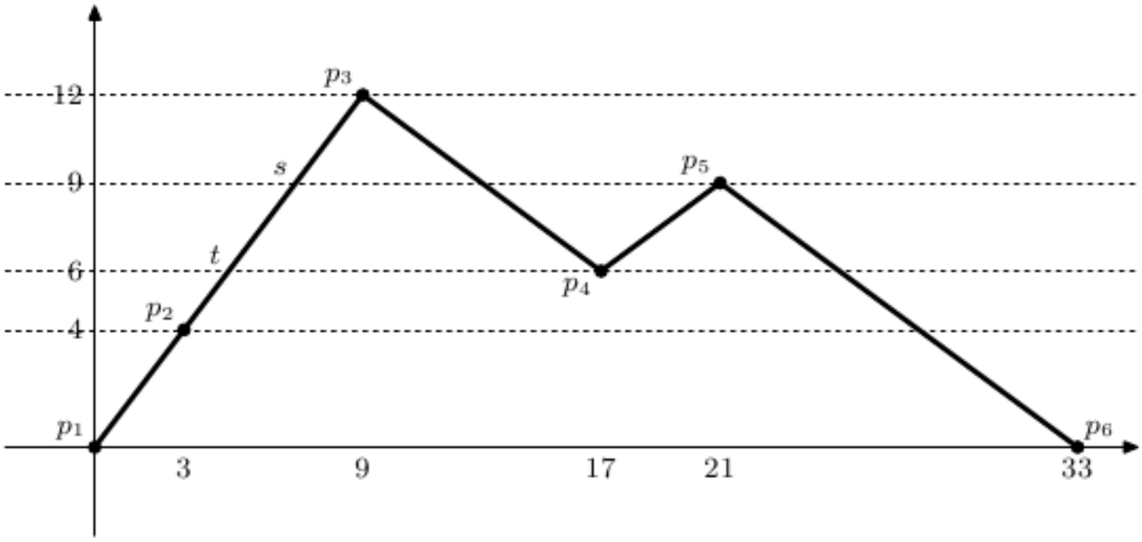


Figure E.1: An illustration of a route

The attempt should be possible for the route as the wise man said. The two climbers, however, could not find a pair of move sequences to achieve the attempt, because they cannot keep their altitudes equal without a complex combination of both forward and backward moves. For example, for the route illustrated above: a climber starting at p_1 (say A) moves to s, and the other climber (say B) moves from p_6 to p_5; then A moves back to t while B moves to p_4; finally A arrives at p_3 and at the same time B also arrives at p_3. Things can be much more complicated and thus they asked you to write a program to find a pair of move sequences for them.

There may exist more than one possible pair of move sequences, and thus you are requested to find the pair of move sequences with the shortest length sum. Here, we measure the length along the route surface, i.e., an uphill path from (0, 0) to (3, 4) has the length of 5.

Input

The input is a sequence of datasets.

The first line of each dataset has an integer indicating the number of points N ($2 \leq N \leq 100$) on the route. Each of the following N lines has the coordinates (x_i, y_i) (i = 1, 2,..., N) of the points: the two start points are (x_1, y_1) and (x_N, y_N); the line segments of the route connect (x_i, y_i) and (x_{i+1}, y_{i+1}) for i = 1, 2,..., N - 1. Here, x_i is the horizontal distance along the route from the start point x_1, and y_i is the altitude relative to the start point y_1. All the coordinates are non-negative integers smaller than 1000, and inequality $x_i < x_{i+1}$ holds for i = 1, 2,..., N - 1, and $0 = y_1 = y_N \leq y_i$ for i = 2, 3,..., N - 1.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output the minimum sum of lengths (along the route) of move sequences until the two climbers meet with each other at a point on the route. Each output value may not have an error greater than 0.01.

Sample Input

```
6
0 0
3 4
9 12
17 6
21 9
33 0
5
0 0
10 0
20 0
30 0
40 0
10
0 0
1 2
3 0
6 3
9 0
11 2
13 0
15 2
16 2
18 0
7
0 0
150 997
300 1
450 999
600 2
750 998
900 0
0
```

Sample Output

```
52.5
40.0
30.3356209304689
10078.072814085803
```

Towns along a Highway

There are several towns on a highway. The highway has no forks. Given the distances between the neighboring towns, we can calculate all distances from any town to others. For example, given five towns (A, B, C, D and E) and the distances between neighboring towns like in Figure C.1, we can calculate the distance matrix as an upper triangular matrix containing the distances between all pairs of the towns, as shown in Figure C.2.

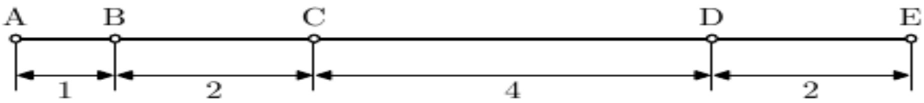


Figure C.1: An example of towns

B	C	D	E	
1	3	7	9	A
	2	6	8	B
		4	6	C
			2	D

Figure C.2: The distance matrix for Figure C.1

In this problem, you must solve the inverse problem. You know only the distances between all pairs of the towns, that is, $N(N - 1)/2$ numbers in the above matrix. Then, you should recover the order of the towns and the distances from each town to the next.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

N
d_1 $d2$... $d_{N(N-1)/2}$

The first line contains an integer N ($2 \le N \le 20$), which is the number of the towns on the highway. The following lines contain $N(N - 1)/2$ integers separated by a single space or a newline, which are the distances between all pairs of the towns. Numbers are given in descending order without any information where they appear in the matrix. You can assume that each distance is between 1 and 400, inclusive. Notice that the longest distance is the distance from the leftmost town to the rightmost.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output N - 1 integers in a line, each of which is the distance from a town to the next in the order of the towns. Numbers in a line must be separated by a single space. If the answer is not unique, output multiple lines ordered by the lexicographical order as a sequence of integers, each of which corresponds to an answer. For example, \``2 10`' should precede \``10 2`'. If no answer exists, output nothing. At the end of the output for each dataset, a line containing five minus symbols \``---`' should be printed for human readability. No extra characters should occur in the output. For example, extra spaces at the end of a line are not allowed.

Sample Input

```
2
1
3
```

```
5 3 2
3
6 3 2
5
9 8 7 6 6 4 3 2 2 1
6
9 8 8 7 6 6 5 5 3 3 3 2 2 1 1
6
11 10 9 8 7 6 6 5 5 4 3 2 2 1 1
7
72 65 55 51 48 45 40 38 34 32 27 25 24 23 21 17 14 13 11 10 7
20
190 189 188 187 186 185 184 183 182 181 180 179 178 177 176 175 174 173 172 171
170 169 168 167 166 165 164 163 162 161 160 159 158 157 156 155 154 153 152 151
150 149 148 147 146 145 144 143 142 141 140 139 138 137 136 135 134 133 132 131
130 129 128 127 126 125 124 123 122 121 120 119 118 117 116 115 114 113 112 111
110 109 108 107 106 105 104 103 102 101 100 99 98 97 96 95 94 93 92 91
90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71
70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51
50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31
30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11
10 9 8 7 6 5 4 3 2 1
19
60 59 58 56 53 52 51 50 48 48 47 46 45 45 44 43 43 42 42 41 41 40 40 40
40 40 40 40 39 39 39 38 38 38 37 37 36 36 35 35 34 33 33 32 32 32 31 31
30 30 30 29 28 28 28 28 27 27 26 26 25 25 25 25 24 24 23 23 23 23 22 22
22 22 21 21 21 21 20 20 20 20 20 20 20 20 20 20 20 20 20 19 19 19 19 19
18 18 18 18 18 17 17 17 17 16 16 16 15 15 15 15 14 14 13 13 13 12 12 12
12 12 11 11 11 10 10 10 10 10 9 9 8 8 8 8 8 8 7 7 7 6 6 6 5 5 5 5 5 5 4
4 4 3 3 3 3 3 3 2 2 2 2 2 2 2 1 1 1 1 1 1
0
```

Sample Output

```
1
-----
2 3
3 2
-----
-----
1 2 4 2
2 4 2 1
----
1 2 3 2 1
-----
1 1 4 2 3
1 5 1 2 2
2 2 1 5 1
3 2 4 1 1
-----
7 14 11 13 10 17
17 10 13 11 14 7
-----
-----
1 1 2 3 5 8 1 1 2 3 5 8 1 1 2 3 5 8
8 5 3 2 1 1 8 5 3 2 1 1 8 5 3 2 1 1
-----
```

Matrix Calculator

Dr. Jimbo, an applied mathematician, needs to calculate matrices all day for solving his own problems. In his laboratory, he uses an excellent application program for manipulating matrix expressions, however, he cannot use it outside his laboratory because the software consumes much of resources. He wants to manipulate matrices outside, so he needs a small program similar to the excellent application for his handheld computer.

Your job is to provide him a program that computes expressions of matrices.

Expressions of matrices are described in a simple language. Its syntax is shown in Table J.1. Note that even a space and a newline have meaningful roles in the syntax.

Table J.1: Syntax in BNF (a newline is denoted by NL)

```
program             ::=  assignment | program assignment
assignment          ::=  var "=" expr "." NL
var                 ::=  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H"
                       | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P"
                       | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X"
                       | "Y" | "Z"
expr                ::=  term | expr "+" term | expr "-" term
term                ::=  factor | term "*" factor
factor              ::=  primary | "-" factor
primary             ::=  inum | var | matrix | "(" expr ")"
                       | indexed-primary | transposed-primary
indexed-primary     ::=  primary "(" expr "," expr ")"
transposed-primary  ::=  primary "'"
matrix              ::=  "[" row-seq "]"
row-seq             ::=  row | row-seq ";" row
row                 ::=  expr | row " " expr
inum                ::=  digit | inum digit
digit               ::=  "0" | "1" | "2" | "3" | "4"
                       | "5" | "6" | "7" | "8" | "9"
```

The start symbol of this syntax is program that is defined as a sequence of assignments in Table J.1. Each assignment has a variable on the left hand side of an equal symbol (``=") and an expression of matrices on the right hand side followed by a period and a newline (NL). It denotes an assignment of the value of the expression to the variable. The variable (var in Table J.1) is indicated by an uppercase Roman letter. The value of the expression (expr) is a matrix or a scalar, whose elements are integers. Here, a scalar integer and a 1×1 matrix whose only element is the same integer can be used interchangeably.

An expression is one or more terms connected by ``+" or ``-" symbols. A term is one or more factors connected by ``*" symbol. These operators (``+", ``-", ``*") are left associative.

A factor is either a primary expression (primary) or a ``-" symbol followed by a factor. This unary operator ``-" is right associative.

The meaning of operators are the same as those in the ordinary arithmetic on matrices: Denoting matrices by A and B, $A + B$, $A - B$, $A*B$, and $-A$ are defined as the matrix sum, difference, product, and negation. The sizes of A and B should be the same for addition and subtraction. The number of columns of A and the number of rows of B should be the same for multiplication.

Note that all the operators +, -, * and unary -- represent computations of addition, subtraction, multiplication and negation modulo $M = 2^{15} = 32768$, respectively. Thus all the values are nonnegative integers between 0 and 32767, inclusive. For example, the result of an expression 2 - 3 should be 32767, instead of -1.

inum is a non-negative decimal integer less than M.

var represents the matrix that is assigned to the variable var in the most recent preceding assignment statement of the same variable.

matrix represents a mathematical matrix similar to a 2-dimensional array whose elements are integers. It is denoted by a row-seq with a pair of enclosing square brackets. row-seq represents a sequence of rows, adjacent two of which are separated by a semicolon. row represents a sequence of expressions, adjacent two of which are separated by a space character.

For example, `[1 2 3;4 5 6]` represents a matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$. The first row has three integers separated by two space characters, i.e. ``1 2 3". The second row has three integers, i.e. ``4 5 6". Here, the `row-seq` consists of the two rows separated by a semicolon. The matrix is denoted by the `row-seq` with a pair of square brackets.

Note that elements of a row may be matrices again. Thus the nested representation of a matrix may appear. The number of rows of the value of each expression of a `row` should be the same, and the number of columns of the value of each `row` of a `row-seq` should be the same.

For example, a matrix represented by

```
[[1 2 3;4 5 6] [7 8;9 10] [11;12];13 14 15 16 17 18]
```

is

$$\begin{pmatrix} 1 & 2 & 3 & 7 & 8 & 11 \\ 4 & 5 & 6 & 9 & 10 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \end{pmatrix}$$

The sizes of matrices should be consistent, as mentioned above, in order to form a well-formed matrix as the result. For example, `[[1 2;3 4] [5;6;7];6 7 8]` is not consistent since the first row ``[1 2;3 4] [5;6;7]" has two matrices (2×2 and 3×1) whose numbers of rows are different. `[1 2;3 4 5]` is not consistent since the number of columns of two rows are different.

The multiplication of 1×1 matrix and $m \times n$ matrix is well-defined for arbitrary $m > 0$ and $n > 0$, since a 1×1 matrices can be regarded as a scalar integer. For example, `2*[1 2;3 4]` and `[1 2;3 4]*3` represent the products of a scalar and a matrix $2*\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$ and $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}*3 = \begin{pmatrix} 3 & 6 \\ 9 & 12 \end{pmatrix}$. `[2]*[1 2;3 4]` and `[1 2;3 4]*[3]` are also well-defined similarly.

An `indexed-primary` is a primary expression followed by two expressions as indices. The first index is $1 \times k$ integer matrix denoted by $(i_1\ i_2 ... i_k)$, and the second index is $1 \times l$ integer matrix denoted by $j_1\ j_2 ... j_l$. The two indices specify the submatrix extracted from the matrix which is the value of the preceding primary expression. The size of the submatrix is $k \times l$ and whose (a, b)-element is the (i_a, j_b)-element of the value of the preceding primary expression. The way of indexing is one-origin, i.e., the first element is indexed by 1.

For example, the value of `([1 2;3 4]+[3 0;0 2])([1],[2])` is equal to 2, since the value of its primary expression is a matrix `[4 2;3 6]`, and the first index `[1]` and the second `[2]` indicate the (1, 2)-element of the matrix. The same integer may appear twice or more in an index matrix, e.g., the first index matrix of an expression `[1 2;3 4]([2 1 1],[2 1])` is `[2 1 1]`, which has two `1`'s. Its value is $\begin{pmatrix} 4 & 3 \\ 2 & 1 \\ 2 & 1 \end{pmatrix}$.

A `transposed-primary` is a primary expression followed by a single quote symbol (``'"), which indicates the transpose operation. The transposed matrix of an $m \times n$ matrix $A = (a_{ij})$ ($i = 1,..., m$ and $j = 1,..., n$) is the $n \times m$ matrix $B = (b_{ij})$ ($i = 1,..., n$ and $j = 1,..., m$), where $b_{ij} = a_{ji}$. For example, the value of `[1 2;3 4]'` is $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$.

Input

The input consists of multiple datasets, followed by a line containing a zero. Each dataset has the following format.

n
program

n is a positive integer, which is followed by a program that is a sequence of single or multiple lines each of which is an assignment statement whose syntax is defined in Table J.1. *n* indicates the number of the assignment statements in the program. All the values of `var`s are undefined at the beginning of a program.

You can assume the following:

- $1 \leq n \leq 10$,
- the number of characters in a line does not exceed 80 (excluding a newline),
- there are no syntax errors and semantic errors (e.g., reference of undefined var),
- the number of rows of matrices appearing in the computations does not exceed 100, and
- the number of columns of matrices appearing in the computations does not exceed 100.

Output

For each dataset, the value of the expression of each assignment statement of the program should be printed in the same order. All the values should be printed as non-negative integers less than M.

When the value is an $m \times n$ matrix $A = (a_{ij})$ ($i = 1,\ldots, m$ and $j = 1,\ldots, n$), m lines should be printed. In the k-th line ($1 \leq k \leq m$), integers of the k-th row, i.e., $a_{k1},\ldots, a_{kn}$, should be printed separated by a space.

After the last value of a dataset is printed, a line containing five minus symbols \`---' should be printed for human readability.

The output should not contain any other extra characters.

Sample Input

```
1
A=[1 2 3;4 5 6].
1
A=[[1 2 3;4 5 6] [7 8;9 10] [11;12];13 14 15 16 17 18].
3
B=[3 -2 1;-9 8 7].
C=([1 2 3;4 5 6]+B)(2,3).
D=([1 2 3;4 5 6]+B)([1 2],[2 3]).
5
A=2*[1 2;-3 4]'.
B=A([2 1 2],[2 1]).
A=[1 2;3 4]*3.
A=[2]*[1 2;3 4].
A=[1 2;3 4]*[3].
2
A=[11 12 13;0 22 23;0 0 33].
A=[A A';--A''' A].
2
A=[1 -1 1;1 1 -1;-1 1 1]*3.
A=[A -A+-A;-A'([3 2 1],[3 2 1]) -A'].
1
A=1([1 1 1],[1 1 1 1]).
3
A=[1 2 -3;4 -5 6;-7 8 9].
B=A([3 1 2],[2 1 3]).
C=A*B-B*A+-A*-B-B*-A.
3
A=[1 2 3 4 5].
B=A'*A.
C=B([1 5],[5 1]).
3
A=[-11 12 13;21 -22 23;31 32 -33].
B=[1 0 0;0 1 0;0 0 1].
C=[(A-B) (A+B)*B (A+B)*(B-A)([1 1 1],[3 2 1]) [1 2 3;2 1 1;-1 2 1]*(A-B)].
3
A=[11 12 13;0 22 23;0 0 33].
B=[1 2].
C=------A((((B))),B)(B,B)''''''.
2
A=1+[2]+[[3]]+[[[4]]]+2*[[[[5]]]]*3.
B=[(-[([(-A)]+-A)])].
8
A=[1 2;3 4].
B=[A A+[1 1;0 1]*4;A+[1 1;0 1]'*8 A+[1 1;0 1]''*12].
C=B([1],[1]).
C=B([1],[1 2 3 4]).
C=B([1 2 3 4],[1]).
C=B([2 3],[2 3]).
A=[1 2;1 2].
D=(A*-A+-A)'(A'(1,[1 2]),A'(2,[1 2])).
0
```

Sample Output

```
1 2 3
4 5 6
-----
1 2 3 7 8 11
4 5 6 9 10 12
13 14 15 16 17 18
-----
3 32766 1
32759 8 7
13
0 4
13 13
-----
2 32762
4 8
8 4
32762 2
8 4
3 6
9 12
2 4
6 8
3 6
9 12
-----
11 12 13
0 22 23
0 0 33
11 12 13 11 0 0
0 22 23 12 22 0
0 0 33 13 23 33
11 0 0 11 12 13
12 22 0 0 22 23
13 23 33 0 0 33
-----
3 32765 3
3 3 32765
32765 3 3
3 32765 3 32762 6 32762
3 3 32765 32762 32762 6
32765 3 3 6 32762 32762
32765 3 32765 32765 32765 3
32765 32765 3 3 32765 32765
3 32765 32765 32765 3 32765
-----
1 1 1 1
1 1 1 1
1 1 1 1
-----
1 2 32765
4 32763 6
32761 8 9
8 32761 9
2 1 32765
32763 4 6
54 32734 32738
32752 32750 174
32598 186 32702
-----
1 2 3 4 5
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
5 1
25 5
-----
32757 12 13
21 32746 23
31 32 32735
1 0 0
0 1 0
0 0 1
32756 12 13 32758 12 13 32573 32588 180 123 62 32725
21 32745 23 21 32747 23 32469 32492 276 28 33 15
31 32 32734 31 32 32736 32365 32396 372 85 32742 32767
-----
11 12 13
0 22 23
0 0 33
1 2
11 12
0 22
```

```
-----
40
80
-----
1 2
3 4
1 2 5 6
3 4 3 8
9 2 13 14
11 12 3 16
1
1 2 5 6
1
3
9
11
4 3
2 13
1 2
1 2
32764 32764
32764 32764
-----
```

Awkward Lights

You are working as a night watchman in an office building. Your task is to check whether all the lights in the building are turned off after all the office workers in the building have left the office. If there are lights that are turned on, you must turn off those lights. This task is not as easy as it sounds to be because of the strange behavior of the lighting system of the building as described below. Although an electrical engineer checked the lighting system carefully, he could not determine the cause of this behavior. So you have no option but to continue to rely on this system for the time being.

Each floor in the building consists of a grid of square rooms. Every room is equipped with one light and one toggle switch. A toggle switch has two positions but they do not mean fixed ON/OFF. When the position of the toggle switch of a room is changed to the other position, in addition to that room, the ON/OFF states of the lights of the rooms at a certain Manhattan distance from that room are reversed. The Manhattan distance between the room at (x_1, y_1) of a grid of rooms and the room at (x_2, y_2) is given by $| x1 - x2| + | y1 - y2|$. For example, if the position of the toggle switch of the room at (2, 2) of a 4 x 4 grid of rooms is changed and the given Manhattan distance is two, the ON/OFF states of the lights of the rooms at (1, 1), (1, 3), (2, 4), (3, 1), (3, 3), and (4, 2) as well as at (2, 2) are reversed as shown in Figure D.1, where black and white squares represent the ON/OFF states of the lights.

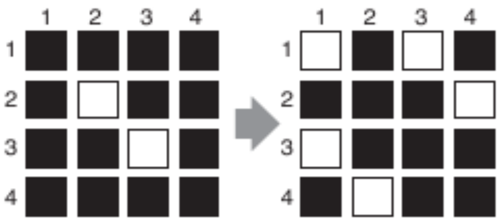


Figure D.1: An example behavior of the lighting system

Your mission is to write a program that answer whether all the lights on a floor can be turned off.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

m n d
S_{11} S_{12} S_{13}...S_{1m}
S_{21} S_{22} S_{23}...S_{2m}

S_{n1} S_{n2} S_{n3}...S_{nm}

The first line of a dataset contains three integers. m and n ($1 \leq m \leq 25$, $1 \leq n \leq 25$) are the numbers of columns and rows of the grid, respectively. d ($1 \leq d \leq m + n$) indicates the Manhattan distance. Subsequent n lines of m integers are the initial ON/OFF states. Each S_{ij} ($1 \leq i \leq n$, $1 \leq j \leq m$) indicates the initial ON/OFF state of the light of the room at (i, j): `` `0' `` for OFF and `` `1' `` for ON.

The end of the input is indicated by a line containing three zeros.

Output

For each dataset, output `` `1' `` if all the lights can be turned off. If not, output `` `0' ``. In either case, print it in one line for each input dataset.

Sample Input

```
1 1 1
1
2 2 1
1 1
1 1
```

```
3 2 1
1 0 1
0 1 0
3 3 1
1 0 1
0 1 0
1 0 1
4 4 2
1 1 0 1
0 0 0 1
1 0 1 1
1 0 0 0
5 5 1
1 1 1 0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
5 5 2
0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
11 11 3
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
11 11 3
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
13 13 7
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
```

Sample Output

```
1
1
0
1
0
0
1
1
0
1
```

Balloon Collecting

``Balloons should be captured efficiently", the game designer says. He is designing an oldfashioned game with two dimensional graphics. In the game, balloons fall onto the ground one after another, and the player manipulates a robot vehicle on the ground to capture the balloons. The player can control the vehicle to move left or right, or simply stay. When one of the balloons reaches the ground, the vehicle and the balloon must reside at the same position, otherwise the balloon will burst and the game ends.

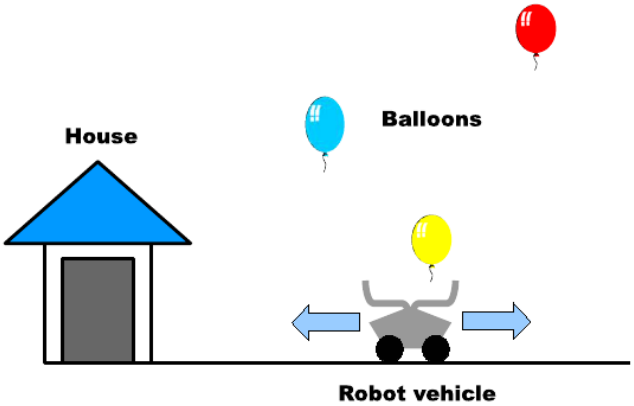


Figure B.1: Robot vehicle and falling balloons

The goal of the game is to store all the balloons into the house at the left end on the game field. The vehicle can carry at most three balloons at a time, but its speed changes according to the number of the carrying balloons. When the vehicle carries k balloons (k = 0, 1, 2, 3), it takes k + 1 units of time to move one unit distance. The player will get higher score when the total moving distance of the vehicle is shorter.

Your mission is to help the game designer check game data consisting of a set of balloons. Given a landing position (as the distance from the house) and a landing time of each balloon, you must judge whether a player can capture all the balloons, and answer the minimum moving distance needed to capture and store all the balloons. The vehicle starts from the house. If the player cannot capture all the balloons, you must identify the first balloon that the player cannot capture.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

n
p_1 t_1

p_n t_n

The first line contains an integer n, which represents the number of balloons ($0 < n \leq 40$). Each of the following n lines contains two integers p_i and t_i ($1 \leq i \leq n$) separated by a space. p_i and t_i represent the position and the time when the i-th balloon reaches the ground ($0 < p_i \leq 100$, $0 < t_i \leq 50000$). You can assume $t_i < t_j$ for $i < j$. The position of the house is 0, and the game starts from the time 0.

The sizes of the vehicle, the house, and the balloons are small enough, and should be ignored. The vehicle needs 0 time for catching the balloons or storing them into the house. The vehicle can start moving immediately after these operations.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output one word and one integer in a line separated by a space. No extra characters should occur in the output.

- If the player can capture all the balloons, output ``OK" and an integer that represents the minimum moving distance of the vehicle to capture and store all the balloons.
- If it is impossible for the player to capture all the balloons, output ``NG" and an integer k such that the k-th balloon in the dataset is the first balloon that the player cannot capture.

Sample Input

```
2
10 100
100 270
2
10 100
100 280
3
100 150
10 360
40 450
3
100 150
10 360
40 440
2
100 10
50 200
2
100 100
50 110
1
15 10
4
1 10
2 20
3 100
90 200
0
```

Sample Output

```
OK 220
OK 200
OK 260
OK 280
NG 1
NG 2
NG 1
OK 188
```

Membership Management

Peter is a senior manager of Agile Change Management (ACM) Inc., where each employee is a member of one or more task groups. Since ACM is agile, task groups are often reorganized and their members frequently change, so membership management is his constant headache.

Peter updates the membership information whenever any changes occur: for instance, the following line written by him means that Carol and Alice are the members of the Design Group.

```
design:carol,alice.
```

The name preceding the colon is the group name and the names following it specify its members.

A smaller task group may be included in a larger one. So, a group name can appear as a member of another group, for instance, as follows.

```
development:alice,bob,design,eve.
```

Simply unfolding the `design` above gives the following membership specification, which is equivalent to the original.

```
development:alice,bob,carol,alice,eve.
```

In this case, however, `alice` occurs twice. After removing one of the duplicates, we have the following more concise specification.

```
development:alice,bob,carol,eve.
```

Your mission in this problem is to write a program that, given group specifications, identifies group members.

Note that Peter's specifications can include deeply nested groups. In the following, for instance, the group one contains a single member `dave`.

```
one:another.
another:yetanother.
yetanother:dave.
```

Input

The input is a sequence of datasets, each being in the following format.

n
$group_1$: $member_{1,1}$,..., $member_{1,m_1}$.

$group_i$: $member_{i,1}$,..., $member_{i,m_i}$.

$group_n$: $member_{n,1}$,..., $member_{n,m_n}$.

The first line contains n, which represents the number of groups and is a positive integer no more than 100. Each of the following n lines contains the membership information of a group: $group_i$ ($1 \leq i \leq n$) is the name of the i-th task group and is followed by a colon (:) and then the list of its m_i *member*s that are delimited by a comma (,) and terminated by a period (.).

Those group names are mutually different. Each m_i ($1 \leq i \leq n$) is between 1 and 10, inclusive. A *member* is another group name if it is one of $group_1$, $group_2$, ..., or $group_n$. Otherwise it is an employee name.

There are no circular (or recursive) definitions of group(s). You may assume that m_i member names of a group are mutually different.

Each group or employee name is a non-empty character string of length between 1 and 15, inclusive, and consists of lowercase letters.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output the number of employees included in the first group of the dataset, that is $group_1$, in a line. No extra characters should occur in the output.

Sample Input

```
2
development:alice,bob,design,eve.
design:carol,alice.
3
one:another.
another:yetanother.
yetanother:dave.
3
friends:alice,bob,bestfriends,carol,fran,badcompany.
bestfriends:eve,alice.
badcompany:dave,carol.
5
a:b,c,d,e.
b:c,d,e,f.
c:d,e,f,g.
d:e,f,g,h.
e:f,g,h,i.
4
aa:bb.
cc:dd,ee.
ff:gg.
bb:cc.
0
```

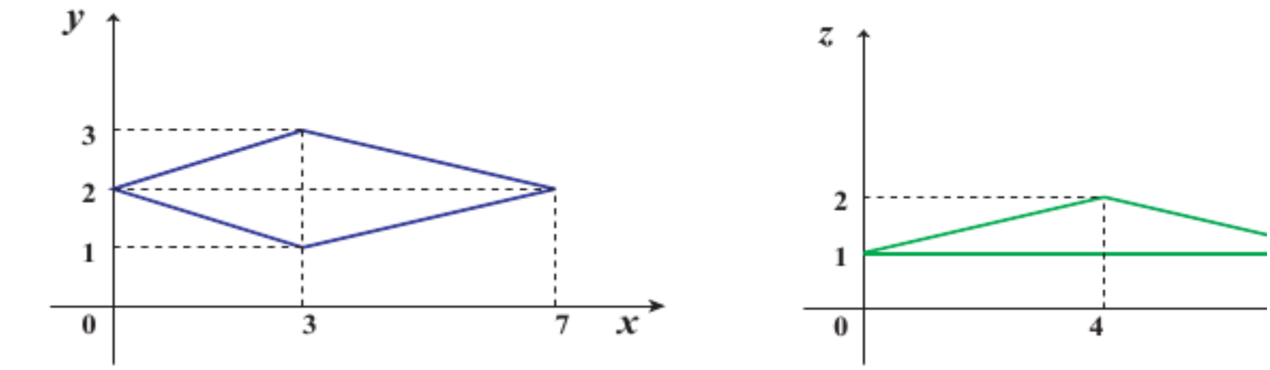
Sample Output

```
4
1
6
4
2
```

Intersection of Two Prisms

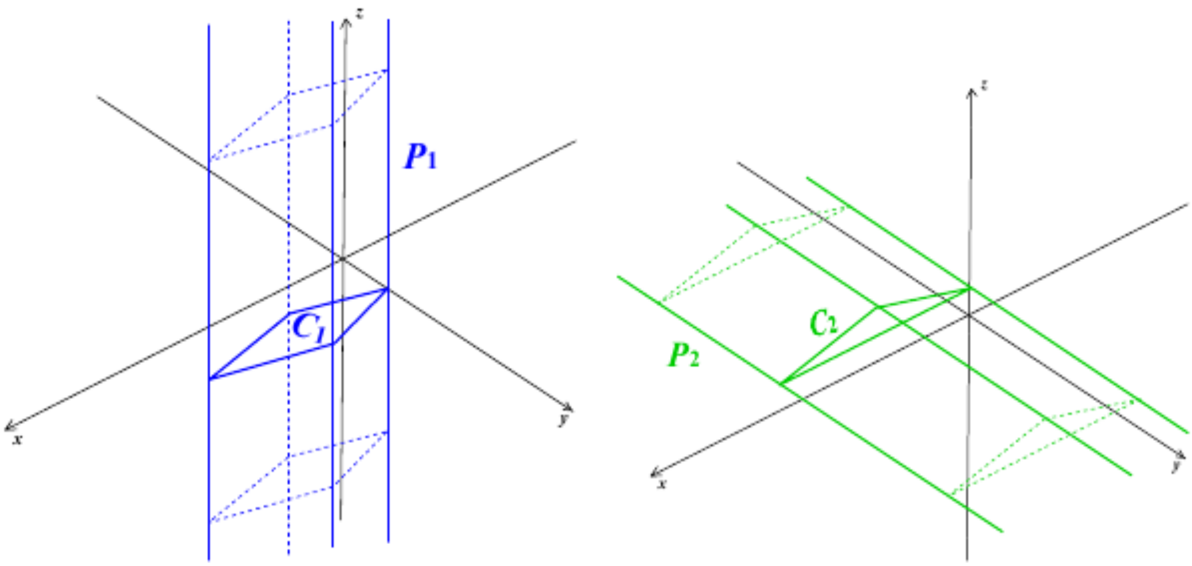
Suppose that P_1 is an infinite-height prism whose axis is parallel to the z-axis, and P_2 is also an infinite-height prism whose axis is parallel to the y-axis. P_1 is defined by the polygon C_1 which is the cross section of P_1 and the xy-plane, and P_2 is also defined by the polygon C_2 which is the cross section of P_2 and the xz-plane.

Figure I.1 shows two cross sections which appear as the first dataset in the sample input, and Figure I.2 shows the relationship between the prisms and their cross sections.



C_1 : Cross section of P_1 and the xy-plane C_2 : Cross section of P_2 and t

Figure I.1: Cross sections of Prisms



P_1 and C_1 P_2 and C_2

Figure I.2: Prisms and their cross sections

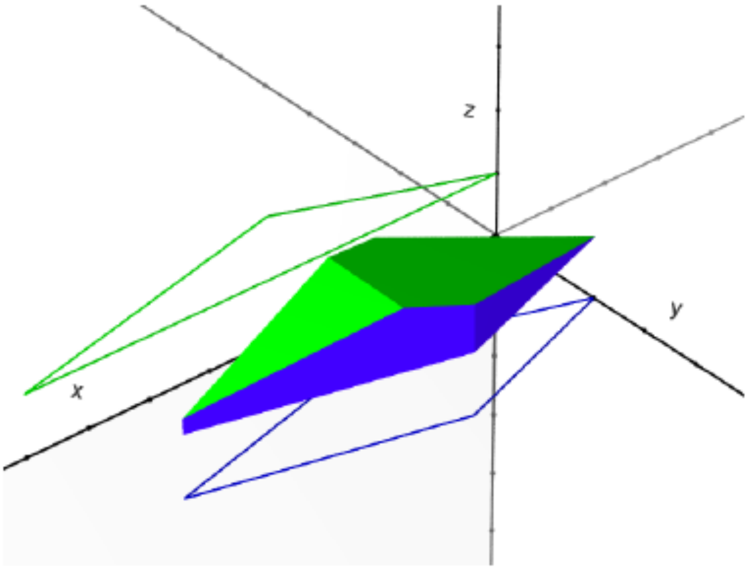


Figure I.3: Intersection of two prisms

Figure I.3 shows the intersection of two prisms in Figure I.2, namely, P_1 and P_2.

Write a program which calculates the volume of the intersection of two prisms.

Input

The input is a sequence of datasets. The number of datasets is less than 200.

Each dataset is formatted as follows.

m n
x_{11} y_{11}
x_{12} y_{12}

x_{1m} y_{1m}
x_{21} z_{21}
x_{22} z_{22}

x_{2n} z_{2n}

m and n are integers ($3 \le m \le 100$, $3 \le n \le 100$) which represent the numbers of the vertices of the polygons, C_1 and C_2, respectively.

x_{1i}, y_{1i}, x_{2j} and z_{2j} are integers between -100 and 100, inclusive. (x_{1i}, y_{1i}) and ($x2j$, $z2j$) mean the i-th and j-th vertices' positions of C_1 and C_2 respectively.

The sequences of these vertex positions are given in the counterclockwise order either on the xy-plane or the xz-plane as in Figure I.1.

You may assume that all the polygons are convex, that is, all the interior angles of the polygons are less than 180 degrees. You may also assume that all the polygons are simple, that is, each polygon's boundary does not cross nor touch itself.

The end of the input is indicated by a line containing two zeros.

Output

For each dataset, output the volume of the intersection of the two prisms, P_1 and P_2, with a decimal representation in a line.

None of the output values may have an error greater than 0.001. The output should not contain any other extra characters.

Sample Input

```
4 3
7 2
3 3
0 2
3 1
4 2
0 1
8 1
4 4
30 2
30 12
2 12
2 2
15 2
30 8
13 14
2 8
8 5
13 5
21 7
21 9
18 15
11 15
6 10
6 8
8 5
10 12
5 9
15 6
20 10
18 12
3 3
5 5
10 3
10 10
20 8
10 15
10 8
4 4
-98 99
-99 -99
99 -98
99 97
-99 99
-98 -98
99 -99
96 99
0 0
```

Sample Output

```
4.708333333333333
1680.0000000000005
491.1500000000007
0.0
7600258.4847715655
```

Find the Multiples

You are given a sequence $a_0 a_1 \cdots a_{N-1}$ of digits and a prime number Q. For each $i \leq j$ with $a_i = 0$, the subsequence $a_i a_{i+1} \cdots a_j$ can be read as a decimal representation of a positive integer. Subsequences with leading zeros are not considered. Your task is to count the number of pairs (i, j) such that the corresponding subsequence is a multiple of Q.

Input

The input consists of at most 50 datasets. Each dataset is represented by a line containing four integers N, S, W, and Q, separated by spaces, where $1 \leq N \leq 10^5$, $1 \leq S \leq 10^9$, $1 \leq W \leq 10^9$, and Q is a prime number less than 10^8 . The sequence $a_0 \cdots a_{N-1}$ of length N is generated by the following code, in which a_i is written as `a[i]`.

```
int g = S;
for(int i=0; i<N; i++) {
      a[i] = (g/7) % 10;
      if( g%2 == 0 ) { g = (g/2); }
      else           { g = (g/2) ^ W; }
}
```

Note: the operators `/`, `%`, and `^` are the integer division, the modulo, and the bitwise exclusiveor, respectively. The above code is meant to be a random number generator. The intended solution does not rely on the way how the sequence is generated.

The end of the input is indicated by a line containing four zeros separated by spaces.

Output

For each dataset, output the answer in a line. You may assume that the answer is less than 2^{30}.

Note:

- In the first dataset, the sequence is 421. We can find two multiples of $Q = 7$, namely, 42 and 21.
- In the second dataset, the sequence is 5052, from which we can find 5, 50, 505, and 5 being the multiples of $Q = 5$. Notice that we don't count 0 or 05 since they are not a valid representation of positive integers. Also notice that we count 5 twice, because it occurs twice in different positions.
- In the third and fourth datasets, the sequences are 95073 and 12221, respectively.

Sample Input

```
3 32 64 7
4 35 89 5
5 555 442 3
5 777 465 11
100000 666 701622763 65537
0 0 0 0
```

Sample Output

```
2
4
6
3
68530
```