Problem A
And Then There Was One
Input: A.in

Let's play a stone removing game.

Initially, n stones are arranged on a circle and numbered $1, \ldots, n$ clockwise (Figure 1). You are also given two numbers k and m. From this state, remove stones one by one following the rules explained below, until only one remains. In step 1, remove stone m. In step 2, locate the k-th next stone clockwise from m and remove it. In subsequent steps, start from the slot of the stone removed in the last step, make k hops clockwise on the remaining stones and remove the one you reach. In other words, skip $(k-1)$ remaining stones clockwise and remove the next one. Repeat this until only one stone is left and answer its number.

For example, the answer for the case $n = 8$, $k = 5$, $m = 3$ is 1, as shown in Figure 1.

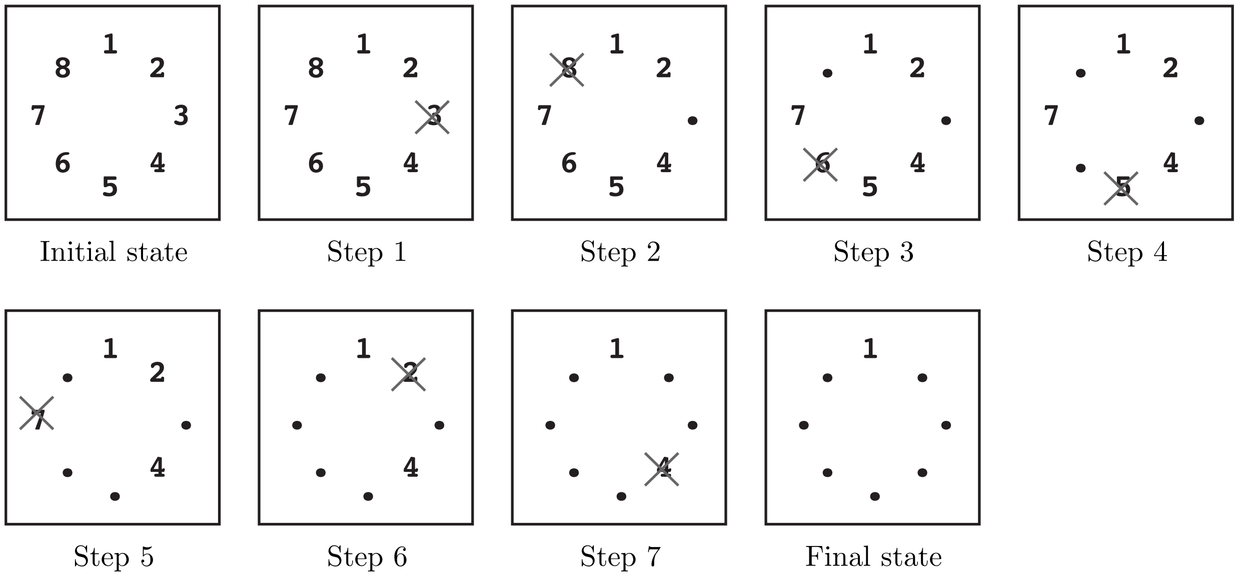

Figure 1: An example game

Initial state: Eight stones are arranged on a circle.

Step 1: Stone 3 is removed since $m = 3$.

Step 2: You start from the slot that was occupied by stone 3. You skip four stones 4, 5, 6 and 7 (since $k = 5$), and remove the next one, which is 8.

Step 3: You skip stones 1, 2, 4 and 5, and thus remove 6. Note that you only count stones that are still on the circle and ignore those already removed. Stone 3 is ignored in this case.

Steps 4–7: You continue until only one stone is left. Notice that in later steps when only a few stones remain, the same stone may be skipped multiple times. For example, stones 1 and 4 are skipped twice in step 7.

Final State: Finally, only one stone, 1, is on the circle. This is the final state, so the answer is 1.

Input

The input consists of multiple datasets each of which is formatted as follows.

$$n \quad k \quad m$$

The last dataset is followed by a line containing three zeros. Numbers in a line are separated by a single space. A dataset satisfies the following conditions.

$$2 \leq n \leq 10000, \quad 1 \leq k \leq 10000, \quad 1 \leq m \leq n$$

The number of datasets is less than 100.

Output

For each dataset, output a line containing the stone number left in the final state. No extra characters such as spaces should appear in the output.

Sample Input

```
8 5 3
100 9999 98
10000 10000 10000
0 0 0
```

Output for the Sample Input

```
1
93
2019
```

Problem B
Prime Gap
Input: B.in

The sequence of $n - 1$ consecutive composite numbers (positive integers that are not prime and not equal to 1) lying between two successive prime numbers p and $p + n$ is called a *prime gap* of length n. For example, $\langle 24, 25, 26, 27, 28 \rangle$ between 23 and 29 is a prime gap of length 6.

Your mission is to write a program to calculate, for a given positive integer k, the length of the prime gap that contains k. For convenience, the length is considered 0 in case no prime gap contains k.

Input

The input is a sequence of lines each of which contains a single positive integer. Each positive integer is greater than 1 and less than or equal to the 100000th prime number, which is 1299709. The end of the input is indicated by a line containing a single zero.

Output

The output should be composed of lines each of which contains a single non-negative integer. It is the length of the prime gap that contains the corresponding positive integer in the input if it is a composite number, or 0 otherwise. No other characters should occur in the output.

Sample Input

```
10
11
27
2
492170
0
```

Output for the Sample Input

```
4
0
6
0
114
```

Problem C
Minimal Backgammon
Input: C.in

Here is a very simple variation of the game backgammon, named "Minimal Backgammon". The game is played by only one player, using only one of the dice and only one checker (the token used by the player).

The game board is a line of $(N + 1)$ squares labeled as 0 (the start) to N (the goal). At the beginning, the checker is placed on the start (square 0). The aim of the game is to bring the checker to the goal (square N). The checker proceeds as many squares as the roll of the dice. The dice generates six integers from 1 to 6 with equal probability.

The checker should not go beyond the goal. If the roll of the dice would bring the checker beyond the goal, the checker retreats from the goal as many squares as the excess. For example, if the checker is placed at the square $(N - 3)$, the roll "5" brings the checker to the square $(N - 2)$, because the excess beyond the goal is 2. At the next turn, the checker proceeds toward the goal as usual.

Each square, except the start and the goal, may be given one of the following two special instructions.

- Lose one turn (labeled "L" in Figure 2)

  If the checker stops here, you cannot move the checker in the next turn.

- Go back to the start (labeled "B" in Figure 2)

  If the checker stops here, the checker is brought back to the start.

Given a game board configuration (the size N, and the placement of the special instructions), you are requested to compute the probability with which the game succeeds within a given number of turns.

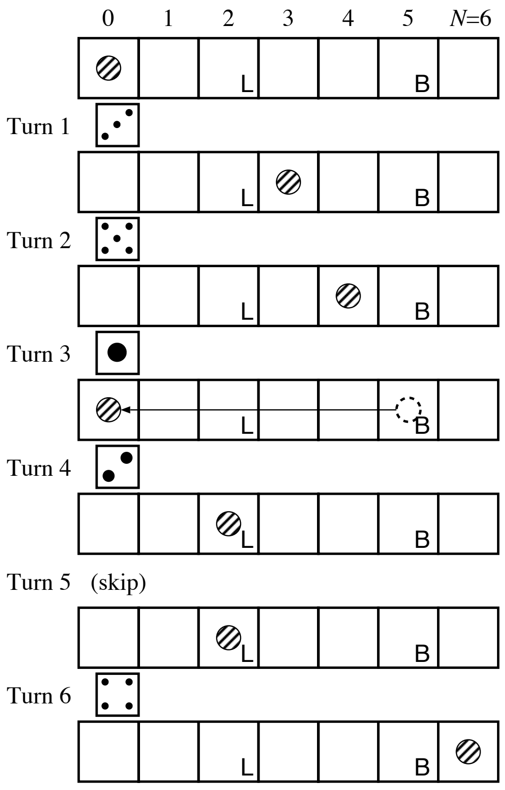

Figure 2: An example game

Input

The input consists of multiple datasets, each containing integers in the following format.

N T L B
$Lose_1$
$\ldots$
$Lose_L$
$Back_1$
$\ldots$
$Back_B$

N is the index of the goal, which satisfies $5 \leq N \leq 100$. T is the number of turns. You are requested to compute the probability of success within T turns. T satisfies $1 \leq T \leq 100$. L is the number of squares marked "Lose one turn", which satisfies $0 \leq L \leq N-1$. B is the number of squares marked "Go back to the start", which satisfies $0 \leq B \leq N-1$. They are separated by a space.

$Lose_i$'s are the indexes of the squares marked "Lose one turn", which satisfy $1 \leq Lose_i \leq N-1$. All $Lose_i$'s are distinct, and sorted in ascending order. $Back_i$'s are the indexes of the squares marked "Go back to the start", which satisfy $1 \leq Back_i \leq N-1$. All $Back_i$'s are distinct, and sorted in ascending order. No numbers occur both in $Lose_i$'s and $Back_i$'s.

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

Output

For each dataset, you should answer the probability with which the game succeeds within the given number of turns. The output should not contain an error greater than 0.00001.

Sample Input

```
6 1 0 0
7 1 0 0
7 2 0 0
6 6 1 1
2
5
7 10 0 6
1
2
3
4
5
6
0 0 0 0
```

Output for the Sample Input

```
0.166667
0.000000
0.166667
0.619642
0.000000
```

Problem D
Lowest Pyramid
Input: D.in

You are constructing a triangular pyramid with a sheet of craft paper with grid lines. Its base and sides are all of triangular shape. You draw the base triangle and the three sides connected to the base on the paper, cut along the outer six edges, fold the edges of the base, and assemble them up as a pyramid.

You are given the coordinates of the base's three vertices, and are to determine the coordinates of the other three. All the vertices must have integral X- and Y-coordinate values between -100 and $+100$ inclusive. Your goal is to minimize the height of the pyramid satisfying these conditions. Figure 3 shows some examples.

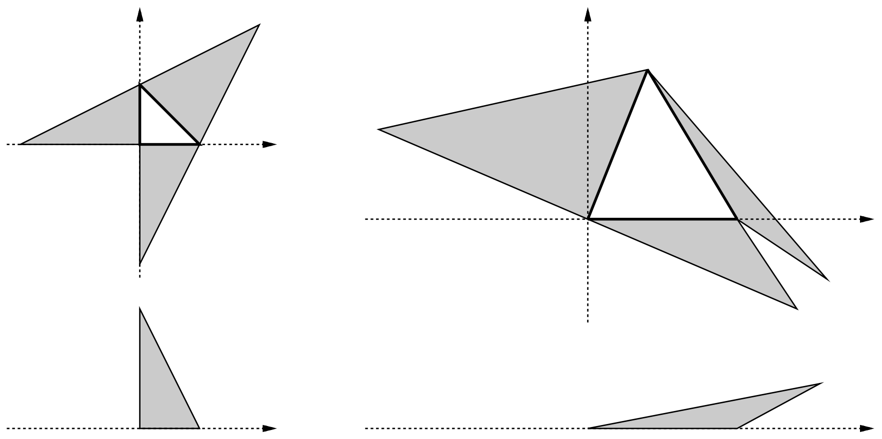

Figure 3: Some craft paper drawings and side views of the assembled pyramids

Input

The input consists of multiple datasets, each in the following format.

$$X_0 \quad Y_0 \quad X_1 \quad Y_1 \quad X_2 \quad Y_2$$

They are all integral numbers between -100 and $+100$ inclusive. (X_0, Y_0), (X_1, Y_1), (X_2, Y_2) are the coordinates of three vertices of the triangular base in counterclockwise order.

The end of the input is indicated by a line containing six zeros separated by a single space.

Output

For each dataset, answer a single number in a separate line. If you can choose three vertices $(X_\mathrm{a}, Y_\mathrm{a})$, $(X_\mathrm{b}, Y_\mathrm{b})$ and $(X_\mathrm{c}, Y_\mathrm{c})$ whose coordinates are all integral values between -100 and $+100$ inclusive, and triangles (X_0, Y_0)–(X_1, Y_1)–$(X_\mathrm{a}, Y_\mathrm{a})$, (X_1, Y_1)–(X_2, Y_2)–$(X_\mathrm{b}, Y_\mathrm{b})$, (X_2, Y_2)–(X_0, Y_0)–$(X_\mathrm{c}, Y_\mathrm{c})$ and (X_0, Y_0)–(X_1, Y_1)–(X_2, Y_2) do not overlap each other (in the XY-plane), and can be assembled as a triangular pyramid of positive (non-zero) height, output the minimum height among such pyramids. Otherwise, output -1.

You may assume that the height is, if positive (non-zero), not less than 0.00001. The output should not contain an error greater than 0.00001.

Sample Input

```
0 0 1 0 0 1
0 0 5 0 2 5
-100 -100 100 -100 0 100
-72 -72 72 -72 0 72
0 0 0 0 0 0
```

Output for the Sample Input

```
2
1.49666
-1
8.52936
```

Problem E
Geometric Map
Input: E.in

Your task in this problem is to create a program that finds the shortest path between two given locations on a given street map, which is represented as a collection of line segments on a plane.

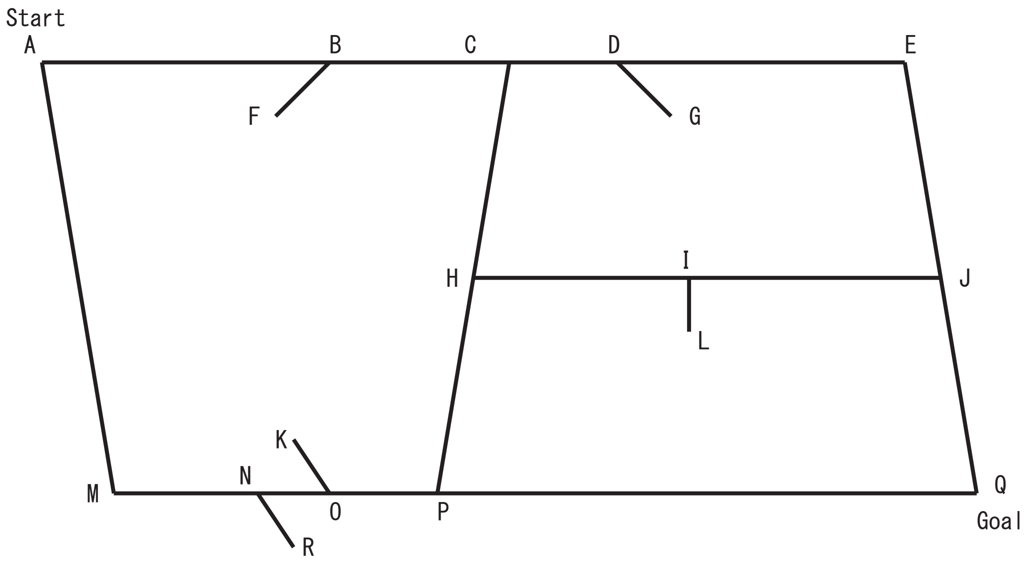

Figure 4: An example map

Figure 4 is an example of a street map, where some line segments represent streets and the others are signs indicating the directions in which cars cannot move. More concretely, `AE`, `AM`, `MQ`, `EQ`, `CP` and `HJ` represent the streets and the others are signs in this map. In general, an end point of a sign touches one and only one line segment representing a street and the other end point is open. Each end point of every street touches one or more streets, but no signs.

The sign `BF`, for instance, indicates that at `B` cars may move left to right but may not in the reverse direction. In general, cars may not move from the obtuse angle side to the acute angle side at a point where a sign touches a street (note that the angle `CBF` is obtuse and the angle `ABF` is acute). Cars may directly move neither from `P` to `M` nor from `M` to `P` since cars moving left to right may not go through `N` and those moving right to left may not go through `O`. In a special case where the angle between a sign and a street is rectangular, cars may not move in either directions at the point. For instance, cars may directly move neither from `H` to `J` nor from `J` to `H`.

You should write a program that finds the shortest path obeying these traffic rules. The length of a line segment between (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Input

The input consists of multiple datasets, each in the following format.

$$\begin{array}{llll} n \\ x_{\rm s} & y_{\rm s} \\ x_{\rm g} & y_{\rm g} \\ x_1^1 & y_1^1 & x_2^1 & y_2^1 \\ \vdots \\ x_1^k & y_1^k & x_2^k & y_2^k \\ \vdots \\ x_1^n & y_1^n & x_2^n & y_2^n \end{array}$$

n, representing the number of line segments, is a positive integer less than or equal to 200.

$(x_{\rm s}, y_{\rm s})$ and $(x_{\rm g}, y_{\rm g})$ are the start and goal points, respectively. You can assume that $(x_{\rm s}, y_{\rm s}) \neq (x_{\rm g}, y_{\rm g})$ and that each of them is located on an end point of some line segment representing a street. You can also assume that the shortest path from $(x_{\rm s}, y_{\rm s})$ to $(x_{\rm g}, y_{\rm g})$ is unique.

(x_1^k, y_1^k) and (x_2^k, y_2^k) are the two end points of the kth line segment. You can assume that $(x_1^k, y_1^k) \neq (x_2^k, y_2^k)$. Two line segments never cross nor overlap. That is, if they share a point, it is always one of their end points.

All the coordinates are non-negative integers less than or equal to 1000. The end of the input is indicated by a line containing a single zero.

Output

For each input dataset, print every *street intersection point* on the shortest path from the start point to the goal point, one in an output line in this order, and a zero in a line following those points. Note that a street intersection point is a point where at least two line segments representing streets meet. An output line for a street intersection point should contain its x- and y-coordinates separated by a space.

Print -1 if there are no paths from the start point to the goal point.

Sample Input

```
8
1 1
4 4
1 1 4 1
1 1 1 4
3 1 3 4
4 3 5 3
```

```
2 4 3 5
4 1 4 4
3 3 2 2
1 4 4 4
9
1 5
5 1
5 4 5 1
1 5 1 1
1 5 5 1
2 3 2 4
5 4 1 5
3 2 2 1
4 2 4 1
1 1 5 1
5 3 4 3
11
5 5
1 0
3 1 5 1
4 3 4 2
3 1 5 5
2 3 2 2
1 0 1 2
1 2 3 4
3 4 5 5
1 0 5 2
4 0 4 1
5 5 5 1
2 3 2 4
0
```

Output for the Sample Input

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

Problem F
Slim Span
Input: F.in

Given an undirected weighted graph G, you should find one of spanning trees specified as follows.

The graph G is an ordered pair (V, E), where V is a set of vertices $\{v_1, v_2, \ldots, v_n\}$ and E is a set of undirected edges $\{e_1, e_2, \ldots, e_m\}$. Each edge $e \in E$ has its weight $w(e)$.

A spanning tree T is a tree (a connected subgraph without cycles) which connects all the n vertices with $n-1$ edges. The *slimness* of a spanning tree T is defined as the difference between the largest weight and the smallest weight among the $n-1$ edges of T.

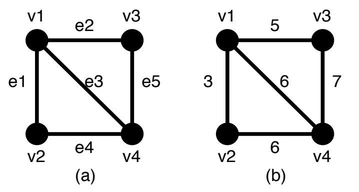

Figure 5: A graph G and the weights of the edges

For example, a graph G in Figure 5(a) has four vertices $\{v_1, v_2, v_3, v_4\}$ and five undirected edges $\{e_1, e_2, e_3, e_4, e_5\}$. The weights of the edges are $w(e_1) = 3$, $w(e_2) = 5$, $w(e_3) = 6$, $w(e_4) = 6$, $w(e_5) = 7$ as shown in Figure 5(b).

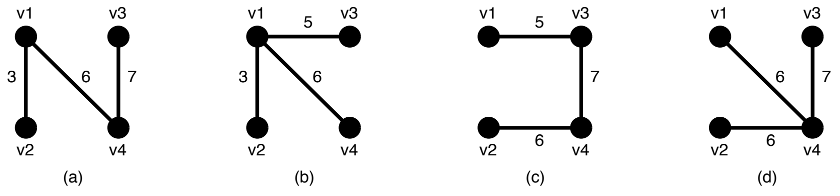

Figure 6: Examples of the spanning trees of G

There are several spanning trees for G. Four of them are depicted in Figure 6(a)∼(d). The spanning tree T_{a} in Figure 6(a) has three edges whose weights are 3, 6 and 7. The largest weight is 7 and the smallest weight is 3 so that the slimness of the tree T_{a} is 4. The slimnesses of spanning trees T_{b}, T_{c} and T_{d} shown in Figure 6(b), (c) and (d) are 3, 2 and 1, respectively. You can easily see the slimness of any other spanning tree is greater than or equal to 1, thus the spanning tree T_{d} in Figure 6(d) is one of the slimmest spanning trees whose slimness is 1.

Your job is to write a program that computes the smallest slimness.

Input

The input consists of multiple datasets, followed by a line containing two zeros separated by a space. Each dataset has the following format.

$$\begin{array}{lll} n & m & \\ a_1 & b_1 & w_1 \\ & \vdots & \\ a_m & b_m & w_m \end{array}$$

Every input item in a dataset is a non-negative integer. Items in a line are separated by a space.

n is the number of the vertices and m the number of the edges. You can assume $2 \leq n \leq 100$ and $0 \leq m \leq n(n-1)/2$. a_k and b_k ($k = 1, \ldots, m$) are positive integers less than or equal to n, which represent the two vertices v_{a_k} and v_{b_k} connected by the kth edge e_k. w_k is a positive integer less than or equal to 10000, which indicates the weight of e_k. You can assume that the graph $G = (V, E)$ is simple, that is, there are no self-loops (that connect the same vertex) nor parallel edges (that are two or more edges whose both ends are the same two vertices).

Output

For each dataset, if the graph has spanning trees, the smallest slimness among them should be printed. Otherwise, -1 should be printed. An output should not contain extra characters.

Sample Input

```
4 5
1 2 3
1 3 5
1 4 6
2 4 6
3 4 7
4 6
1 2 10
1 3 100
1 4 90
2 3 20
2 4 80
3 4 40
2 1
1 2 1
3 0
3 1
1 2 1
3 3
1 2 2
```

```
2 3 5
1 3 6
5 10
1 2 110
1 3 120
1 4 130
1 5 120
2 3 110
2 4 120
2 5 130
3 4 120
3 5 110
4 5 120
5 10
1 2 9384
1 3 887
1 4 2778
1 5 6916
2 3 7794
2 4 8336
2 5 5387
3 4 493
3 5 6650
4 5 1422
5 8
1 2 1
2 3 100
3 4 100
4 5 100
1 5 50
2 5 50
3 5 50
4 1 150
0 0
```

Output for the Sample Input

```
1
20
0
-1
-1
1
0
1686
50
```

Problem G
The Morning after Halloween
Input: G.in

You are working for an amusement park as an operator of an *obakeyashiki*, or a haunted house, in which guests walk through narrow and dark corridors. The house is proud of their lively ghosts, which are actually robots remotely controlled by the operator, hiding here and there in the corridors. One morning, you found that the ghosts are not in the positions where they are supposed to be. Ah, yesterday was Halloween. Believe or not, paranormal spirits have moved them around the corridors in the night. You have to move them into their right positions before guests come. Your manager is eager to know how long it takes to restore the ghosts.

In this problem, you are asked to write a program that, given a floor map of a house, finds the smallest number of steps to move all ghosts to the positions where they are supposed to be.

A floor consists of a matrix of square cells. A cell is either a wall cell where ghosts cannot move into or a corridor cell where they can.

At each step, you can move any number of ghosts simultaneously. Every ghost can either stay in the current cell, or move to one of the corridor cells in its 4-neighborhood (i.e. immediately left, right, up or down), if the ghosts satisfy the following conditions:

1. No more than one ghost occupies one position at the end of the step.
2. No pair of ghosts exchange their positions one another in the step.

For example, suppose ghosts are located as shown in the following (partial) map, where a sharp sign (‘`#`’) represents a wall cell and ‘`a`’, ‘`b`’, and ‘`c`’ ghosts.

```
####
 ab#
#c##
####
```

The following four maps show the only possible positions of the ghosts after one step.

```
####   ####   ####   ####
 ab#   a b#   acb#   ab #
#c##   #c##   # ##   #c##
####   ####   ####   ####
```

Input

The input consists of at most 10 datasets, each of which represents a floor map of a house. The format of a dataset is as follows.

$$\begin{array}{llll} w & h & n & \\ c_{11} & c_{12} & \cdots & c_{1w} \\ c_{21} & c_{22} & \cdots & c_{2w} \\ \vdots & \vdots & \ddots & \vdots \\ c_{h1} & c_{h2} & \cdots & c_{hw} \end{array}$$

w, h and n in the first line are integers, separated by a space. w and h are the floor width and height of the house, respectively. n is the number of ghosts. They satisfy the following constraints.

$$4 \leq w \leq 16, \qquad 4 \leq h \leq 16, \qquad 1 \leq n \leq 3$$

Subsequent h lines of w characters are the floor map. Each of c_{ij} is either:

- a '`#`' representing a wall cell,
- a lowercase letter representing a corridor cell which is the initial position of a ghost,
- an uppercase letter representing a corridor cell which is the position where the ghost corresponding to its lowercase letter is supposed to be, or
- a space representing a corridor cell that is none of the above.

In each map, each of the first n letters from `a` and the first n letters from `A` appears once and only once. Outermost cells of a map are walls; i.e. all characters of the first and last lines are sharps; and the first and last characters on each line are also sharps. All corridor cells in a map are connected; i.e. given a corridor cell, you can reach any other corridor cell by following corridor cells in the 4-neighborhoods. Similarly, all wall cells are connected. Any 2×2 area on any map has at least one sharp. You can assume that every map has a sequence of moves of ghosts that restores all ghosts to the positions where they are supposed to be.

The last dataset is followed by a line containing three zeros separated by a space.

Output

For each dataset in the input, one line containing the smallest number of steps to restore ghosts into the positions where they are supposed to be should be output. An output line should not contain extra characters such as spaces.

Sample Input

```
5 5 2
#####
#A#B#
#   #
#b#a#
#####
16 4 3
################
## ########## ##
#    ABCcba    #
################
16 16 3
################
### ##    #   ##
##  #  ##   # c#
#  ## ########b#
# ##  # #   #  #
#  # ##   # # ##
##  a#  # # #  #
### ## #### ## #
##   #   #  #  #
#  ##### # ## ##
####   #B# #   #
##  C#   #   ###
#  # # ####### #
# ######  A##  #
#        #    ##
################
0 0 0
```

Output for the Sample Input

```
7
36
77
```

Problem H
Bug Hunt
Input: H.in

In this problem, we consider a simple programming language that has only declarations of one-dimensional integer arrays and assignment statements. The problem is to find a bug in the given program.

The syntax of this language is given in BNF as follows:

⟨*program*⟩	::=	⟨*declaration*⟩ | ⟨*program*⟩⟨*declaration*⟩ | ⟨*program*⟩⟨*assignment*⟩
⟨*declaration*⟩	::=	⟨*array_name*⟩ `[`⟨*number*⟩`]` ⟨*new_line*⟩
⟨*assignment*⟩	::=	⟨*array_name*⟩ `[`⟨*expression*⟩`]=`⟨*expression*⟩⟨*new_line*⟩
⟨*expression*⟩	::=	⟨*number*⟩ | ⟨*array_name*⟩ `[`⟨*expression*⟩`]`
⟨*number*⟩	::=	⟨*digit*⟩ | ⟨*digit_positive*⟩⟨*digit_string*⟩
⟨*digit_string*⟩	::=	⟨*digit*⟩ | ⟨*digit*⟩⟨*digit_string*⟩
⟨*digit_positive*⟩	::=	`1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9`
⟨*digit*⟩	::=	`0` | ⟨*digit_positive*⟩
⟨*array_name*⟩	::=	`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` |
		`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`

where ⟨*new_line*⟩ denotes a new line character (LF).

Characters used in a program are alphabetical letters, decimal digits, `=`, `[`, `]` and new line characters. No other characters appear in a program.

A declaration declares an array and specifies its length. Valid indices of an array of length n are integers between 0 and $n-1$, inclusive. Note that the array names are case sensitive, i.e. array `a` and array `A` are different arrays. The initial value of each element in the declared array is undefined.

For example, array `a` of length 10 and array `b` of length 5 are declared respectively as follows.

```
a[10]
b[5]
```

An expression evaluates to a non-negative integer. A ⟨*number*⟩ is interpreted as a decimal integer. An ⟨*array_name*⟩`[`⟨*expression*⟩`]` evaluates to the value of the ⟨*expression*⟩-th element of the array. An assignment assigns the value denoted by the right hand side to the array element specified by the left hand side.

Examples of assignments are as follows.

```
a[0]=3
a[1]=0
a[2]=a[a[1]]
a[a[0]]=a[1]
```

A program is executed from the first line, line by line. You can assume that an array is declared once and only once before any of its element is assigned or referred to.

Given a program, you are requested to find the following bugs.

- An index of an array is invalid.
- An array element that has not been assigned before is referred to in an assignment as an index of array or as the value to be assigned.

You can assume that other bugs, such as syntax errors, do not appear. You can also assume that integers represented by ⟨*number*⟩s are between 0 and $2^{31} - 1$ (= 2147483647), inclusive.

Input

The input consists of multiple datasets followed by a line which contains only a single '.' (period). Each dataset consists of a program also followed by a line which contains only a single '.' (period). A program does not exceed 1000 lines. Any line does not exceed 80 characters excluding a new line character.

Output

For each program in the input, you should answer the line number of the assignment in which the first bug appears. The line numbers start with 1 for each program. If the program does not have a bug, you should answer zero. The output should not contain extra characters such as spaces.

Sample Input

```
a[3]
a[0]=a[1]
.
x[1]
x[0]=x[0]
.
a[0]
a[0]=1
.
b[2]
b[0]=2
b[1]=b[b[0]]
b[0]=b[1]
.
g[2]
G[10]
g[0]=0
g[1]=G[0]
.
a[2147483647]
a[0]=1
B[2]
B[a[0]]=2
a[B[a[0]]]=3
a[2147483646]=a[2]
.
.
```

Output for the Sample Input

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

Problem I
Most Distant Point from the Sea
Input: I.in

The main land of Japan called Honshu is an island surrounded by the sea. In such an island, it is natural to ask a question: "Where is the most distant point from the sea?" The answer to this question for Honshu was found in 1996. The most distant point is located in former Usuda Town, Nagano Prefecture, whose distance from the sea is 114.86 km.

In this problem, you are asked to write a program which, given a map of an island, finds the most distant point from the sea in the island, and reports its distance from the sea. In order to simplify the problem, we only consider maps representable by convex polygons.

Input

The input consists of multiple datasets. Each dataset represents a map of an island, which is a convex polygon. The format of a dataset is as follows.

$$\begin{array}{ll} n & \\ x_1 & y_1 \\ & \vdots \\ x_n & y_n \end{array}$$

Every input item in a dataset is a non-negative integer. Two input items in a line are separated by a space.

n in the first line is the number of vertices of the polygon, satisfying $3 \le n \le 100$. Subsequent n lines are the x- and y-coordinates of the n vertices. Line segments $(x_i,\ y_i)$–$(x_{i+1},\ y_{i+1})$ $(1 \le i \le n-1)$ and the line segment $(x_n,\ y_n)$–$(x_1,\ y_1)$ form the border of the polygon in counterclockwise order. That is, these line segments see the inside of the polygon in the left of their directions. All coordinate values are between 0 and 10000, inclusive.

You can assume that the polygon is simple, that is, its border never crosses or touches itself. As stated above, the given polygon is always a convex one.

The last dataset is followed by a line containing a single zero.

Output

For each dataset in the input, one line containing the distance of the most distant point from the sea should be output. An output line should not contain extra characters such as spaces.

The answer should not have an error greater than 0.00001 (10^{-5}). You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.

Sample Input

```
4
0 0
10000 0
10000 10000
0 10000
3
0 0
10000 0
7000 1000
6
0 40
100 20
250 40
250 70
100 90
0 70
3
0 0
10000 10000
5000 5001
0
```

Output for the Sample Input

```
5000.000000
494.233641
34.542948
0.353553
```

Problem J
The Teacher's Side of Math
Input: J.in

One of the tasks students routinely carry out in their mathematics classes is to solve a polynomial equation. It is, given a polynomial, say $X^2 - 4X + 1$, to find its roots $(2 \pm \sqrt{3})$.

If the students' task is to find the roots of a given polynomial, the teacher's task is then to find a polynomial that has a given root. Ms. Galsone is an enthusiastic mathematics teacher who is bored with finding solutions of quadratic equations that are as simple as $a + b\sqrt{c}$. She wanted to make higher-degree equations whose solutions are a little more complicated. As usual in problems in mathematics classes, she wants to maintain all coefficients to be integers and keep the degree of the polynomial as small as possible (provided it has the specified root). Please help her by writing a program that carries out the task of the teacher's side.

You are given a number t of the form:

$$t = \sqrt[m]{a} + \sqrt[n]{b}$$

where a and b are distinct prime numbers, and m and n are integers greater than 1.

In this problem, you are asked to find t's *minimal polynomial on integers*, which is the polynomial $F(X) = a_dX^d + a_{d-1}X^{d-1} + \cdots + a_1X + a_0$ satisfying the following conditions.

1. Coefficients $a_0, \ldots, a_d$ are integers and $a_d > 0$.
2. $F(t) = 0$.
3. The degree d is minimum among polynomials satisfying the above two conditions.
4. $F(X)$ is *primitive*. That is, coefficients $a_0, \ldots, a_d$ have no common divisors greater than one.

For example, the minimal polynomial of $\sqrt{3}+\sqrt{2}$ on integers is $F(X) = X^4 - 10X^2 + 1$. Verifying $F(t) = 0$ is as simple as the following ($\alpha = \sqrt{3}$, $\beta = \sqrt{2}$).

$$\begin{aligned}
F(t) &= (\alpha + \beta)^4 - 10(\alpha + \beta)^2 + 1 \\
&= (\alpha^4 + 4\alpha^3\beta + 6\alpha^2\beta^2 + 4\alpha\beta^3 + \beta^4) - 10(\alpha^2 + 2\alpha\beta + \beta^2) + 1 \\
&= 9 + 12\alpha\beta + 36 + 8\alpha\beta + 4 - 10(3 + 2\alpha\beta + 2) + 1 \\
&= (9 + 36 + 4 - 50 + 1) + (12 + 8 - 20)\alpha\beta \\
&= 0
\end{aligned}$$

Verifying that the degree of $F(t)$ is in fact minimum is a bit more difficult. Fortunately, under the condition given in this problem, which is that a and b are distinct prime numbers and m and n greater than one, the degree of the minimal polynomial is always mn. Moreover, it is always *monic*. That is, the coefficient of its highest-order term (a_d) is one.

Input

The input consists of multiple datasets, each in the following format.

$$a \quad m \quad b \quad n$$

This line represents $\sqrt[m]{a} + \sqrt[n]{b}$. The last dataset is followed by a single line consisting of four zeros. Numbers in a single line are separated by a single space.

Every dataset satisfies the following conditions.

1. $\sqrt[m]{a} + \sqrt[n]{b} \leq 4$.
2. $mn \leq 20$.
3. The coefficients of the answer $a_0, \ldots, a_d$ are between $(-2^{31}+1)$ and $(2^{31}-1)$, inclusive.

Output

For each dataset, output the coefficients of its minimal polynomial on integers $F(X) = a_d X^d + a_{d-1}X^{d-1} + \cdots + a_1 X + a_0$, in the following format.

$$a_d \quad a_{d-1} \quad \ldots \quad a_1 \quad a_0$$

Non-negative integers must be printed without a sign (+ or −). Numbers in a single line must be separated by a single space and no other characters or extra spaces may appear in the output.

Sample Input

```
3 2 2 2
3 2 2 3
2 2 3 4
31 4 2 3
3 2 2 7
0 0 0 0
```

Output for the Sample Input

```
1 0 -10 0 1
1 0 -9 -4 27 -36 -23
1 0 -8 0 18 0 -104 0 1
1 0 0 -8 -93 0 24 -2976 2883 -32 -3720 -23064 -29775
1 0 -21 0 189 0 -945 -4 2835 -252 -5103 -1260 5103 -756 -2183
```