Problem I

~~Problem A~~

How I Wonder What You Are!

Input: A.txt

One of the questions children often ask is "How many stars are there in the sky?" Under ideal conditions, even with the naked eye, nearly eight thousands are observable in the northern hemisphere. With a decent telescope, you may find many more, but, as the sight field will be limited, you may find much less at a time.

Children may ask the same questions to their parents on a planet of some solar system billions of light-years away from the Earth. Their telescopes are similar to ours with circular sight fields, but alien kids have many eyes and can look into different directions at a time through many telescopes.

Given a set of positions of stars, a set of telescopes and the directions they are looking to, your task is to count up how many stars can be seen through these telescopes.

Input

The input consists of one or more datasets. The number of datasets is less than 50. Each dataset describes stars and the parameters of the telescopes used.

The first line of a dataset contains a positive integer n not exceeding 500, meaning the number of stars. Each of the n lines following it contains three decimal fractions, s_{x}, s_{y}, and s_{z}. They give the position $(s_{\mathrm{x}}, s_{\mathrm{y}}, s_{\mathrm{z}})$ of the star described in Euclidean coordinates. You may assume $-1000 \leq s_{\mathrm{x}} \leq 1000$, $-1000 \leq s_{\mathrm{y}} \leq 1000$, $-1000 \leq s_{\mathrm{z}} \leq 1000$ and $(s_{\mathrm{x}}, s_{\mathrm{y}}, s_{\mathrm{z}}) \neq (0,0,0)$.

Then comes a line containing a positive integer m not exceeding 50, meaning the number of telescopes. Each of the following m lines contains four decimal fractions, t_{x}, t_{y}, t_{z}, and ψ, describing a telescope.

The first three numbers represent the direction of the telescope. All the telescopes are at the origin of the coordinate system $(0,0,0)$ (we ignore the size of the planet). The three numbers give the point $(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}})$ which can be seen in the center of the sight through the telescope. You may assume $-1000 \leq t_{\mathrm{x}} \leq 1000$, $-1000 \leq t_{\mathrm{y}} \leq 1000$, $-1000 \leq t_{\mathrm{z}} \leq 1000$ and $(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}}) \neq (0,0,0)$.

The fourth number ψ $(0 \leq \psi \leq \pi/2)$ gives the angular radius, in radians, of the sight field of the telescope.

Let us define that $\theta_{i,j}$ is the angle between the direction of the i-th star and the center direction of the j-th telescope and ψ_j is the angular radius of the sight field of the j-th telescope. The i-th star is observable through the j-th telescope if and only if $\theta_{i,j}$ is less than ψ_j. You may assume that $|\theta_{i,j} - \psi_j| > 0.00000001$ for all pairs of i and j.

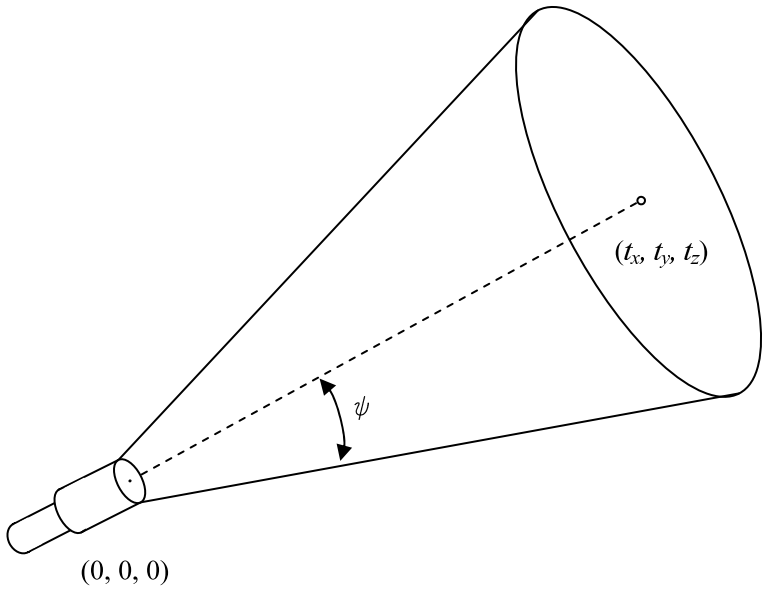

Figure 1: Direction and angular radius of a telescope

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

Output

For each dataset, one line containing an integer meaning the number of stars observable through the telescopes should be output. No other characters should be contained in the output. Note that stars that can be seen through more than one telescope should not be counted twice or more.

Sample Input

```
3
100 0 500
-500.243 -200.1 -300.5
0 300 200
2
1 1 1 0.65
-1 0 0 1.57
3
1 0 0
0 1 0
0 0 1
4
1 -1 -1 0.9553
-1 1 -1 0.9554
```

```
-1 -1 1 0.9553
-1 1 -1 0.9554
3
1 0 0
0 1 0
0 0 1
4
1 -1 -1 0.9553
-1 1 -1 0.9553
-1 -1 1 0.9553
-1 1 -1 0.9553
0
```

Output for the Sample Input

```
2
1
0
```

Problem J

~~Problem B~~

How I Mathematician Wonder What You Are!
Input: B.txt

After counting so many stars in the sky in his childhood, Isaac, now an astronomer and a mathematician, uses a big astronomical telescope and lets his image processing program count stars. The hardest part of the program is to judge if a shining object in the sky is really a star. As a mathematician, the only way he knows is to apply a mathematical definition of *stars*.

The mathematical definition of a star shape is as follows: A planar shape F is *star-shaped* if and only if there is a point C $\in F$ such that, for any point P $\in F$, the line segment CP is contained in F. Such a point C is called a *center* of F. To get accustomed to the definition, let's see some examples below.

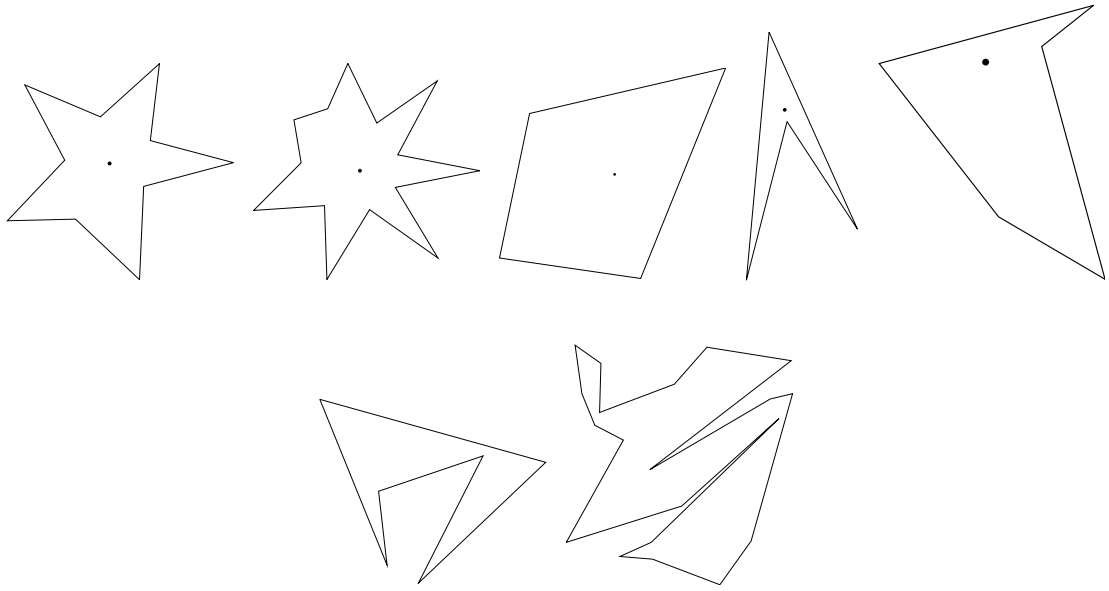

Figure 2: Star shapes (the first row) and non-star shapes (the second row)

The first two are what you would normally call stars. According to the above definition, however, all shapes in the first row are star-shaped. The two in the second row are not. For each star shape, a center is indicated with a dot. Note that a star shape in general has infinitely many centers. For example, for the third quadrangular shape, all points in it are centers.

Your job is to write a program that tells whether a given polygonal shape is star-shaped or not.

Input

The input is a sequence of datasets followed by a line containing a single zero. Each dataset specifies a polygon, and is formatted as follows.

$$
\begin{array}{ll}
n & \\
x_1 & y_1 \\
x_2 & y_2 \\
\multicolumn{2}{c}{\vdots} \\
x_n & y_n
\end{array}
$$

The first line is the number of vertices, n, which satisfies $4 \leq n \leq 50$. Subsequent n lines are the x- and y-coordinates of the n vertices. They are integers and satisfy $0 \leq x_i \leq 10000$ and $0 \leq y_i \leq 10000$ $(i = 1, \ldots, n)$. Line segments (x_i, y_i)–(x_{i+1}, y_{i+1}) $(i = 1, \ldots, n-1)$ and the line segment (x_n, y_n)–(x_1, y_1) form the border of the polygon in the counterclockwise order. That is, these line segments see the inside of the polygon in the left of their directions.

You may assume that the polygon is *simple*, that is, its border never crosses or touches itself. You may also assume that no three edges of the polygon meet at a single point even when they are infinitely extended.

Output

For each dataset, output "1" if the polygon is star-shaped and "0" otherwise. Each number must be in a separate line and the line should not contain any other characters.

Sample Input

```
6
66 13
96 61
76 98
13 94
4 0
45 68
8
27 21
55 14
93 12
56 95
15 48
38 46
51 65
64 31
0
```

Output for the Sample Input

```
1
0
```

Problem K

~~Problem C~~
Cubic Eight-Puzzle
Input: C.txt

Let's play a puzzle using eight cubes placed on a 3×3 board leaving one empty square.

Faces of cubes are painted with three colors. As a puzzle step, you can roll one of the cubes to the adjacent empty square. Your goal is to make the specified color pattern visible from above by a number of such steps.

The rules of this puzzle are as follows.

1. **Coloring of Cubes:** All the cubes are colored in the same way as shown in Figure 3. The opposite faces have the same color.

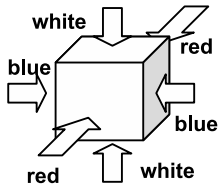

Figure 3: Coloring of a cube

2. **Initial Board State:** Eight cubes are placed on the 3×3 board leaving one empty square. All the cubes have the same orientation as shown in Figure 4. As shown in the figure, squares on the board are given x and y coordinates, (1, 1), (1, 2), ..., and (3, 3). The position of the initially empty square may vary.

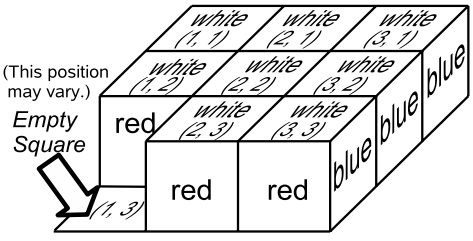

Figure 4: Initial board state

3. **Rolling Cubes:** At each step, we can choose one of the cubes adjacent to the empty

square and roll it into the empty square, leaving the original position empty. Figure 5 shows an example.

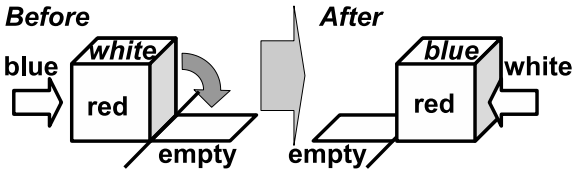

Figure 5: Rolling a cube

4. **Goal:** The goal of this puzzle is to arrange the cubes so that their top faces form the specified color pattern by a number of cube rolling steps described above.

Your task is to write a program that finds the minimum number of steps required to make the specified color pattern from the given initial state.

Input

The input is a sequence of datasets. The end of the input is indicated by a line containing two zeros separated by a space. The number of datasets is less than 16. Each dataset is formatted as follows.

x y
F_{11} F_{21} F_{31}
F_{12} F_{22} F_{32}
F_{13} F_{23} F_{33}

The first line contains two integers x and y separated by a space, indicating the position (x, y) of the initially empty square. The values of x and y are 1, 2, or 3.

The following three lines specify the color pattern to make. Each line contains three characters F_{1j}, F_{2j}, and F_{3j}, separated by a space. Character F_{ij} indicates the top color of the cube, if any, at position (i, j) as follows:

`B`: Blue,

`W`: White,

`R`: Red,

`E`: the square is Empty.

There is exactly one '`E`' character in each dataset.

Output

For each dataset, output the minimum number of steps to achieve the goal, when the goal can be reached within 30 steps. Otherwise, output "-1" for the dataset.

Sample Input

```
1 2
W W W
E W W
W W W
2 1
R B W
R W W
E W W
3 3
W B W
B R E
R B R
3 3
B W R
B W R
B E R
2 1
B B B
B R B
B R E
1 1
R R R
W W W
R R E
2 1
R R R
B W B
R R E
3 2
R R R
W E W
R R R
0 0
```

Output for the Sample Input

```
0
3
13
23
29
30
-1
-1
```

Problem L

~~Problem D~~

Sum of Different Primes
Input: D.txt

A positive integer may be expressed as a sum of different prime numbers (primes), in one way or another. Given two positive integers n and k, you should count the number of ways to express n as a sum of k different primes. Here, two ways are considered to be the same if they sum up the same set of the primes. For example, 8 can be expressed as $3+5$ and $5+3$ but they are not distinguished.

When n and k are 24 and 3 respectively, the answer is two because there are two sets $\{2,3,19\}$ and $\{2,5,17\}$ whose sums are equal to 24. There are no other sets of three primes that sum up to 24. For $n = 24$ and $k = 2$, the answer is three, because there are three sets $\{5,19\}$, $\{7,17\}$ and $\{11,13\}$. For $n = 2$ and $k = 1$, the answer is one, because there is only one set $\{2\}$ whose sum is 2. For $n = 1$ and $k = 1$, the answer is zero. As 1 is not a prime, you shouldn't count $\{1\}$. For $n = 4$ and $k = 2$, the answer is zero, because there are no sets of two different primes whose sums are 4.

Your job is to write a program that reports the number of such ways for the given n and k.

Input

The input is a sequence of datasets followed by a line containing two zeros separated by a space. A dataset is a line containing two positive integers n and k separated by a space. You may assume that $n \leq 1120$ and $k \leq 14$.

Output

The output should be composed of lines, each corresponding to an input dataset. An output line should contain one non-negative integer indicating the number of ways for n and k specified in the corresponding dataset. You may assume that it is less than 2^{31}.

Sample Input

```
24 3
24 2
2 1
1 1
4 2
18 3
17 1
17 3
17 4
100 5
1000 10
1120 14
0 0
```

Output for the Sample Input

```
2
3
1
0
0
2
1
0
1
55
200102899
2079324314
```

Problem M

~~Problem E~~
Manhattan Wiring
Input: E.txt

There is a rectangular area containing $n \times m$ cells. Two cells are marked with "2", and another two with "3". Some cells are occupied by obstacles. You should connect the two "2"s and also the two "3"s with non-intersecting lines. Lines can run only vertically or horizontally connecting centers of cells without obstacles.

Lines cannot run on a cell with an obstacle. Only one line can run on a cell at most once. Hence, a line cannot intersect with the other line, nor with itself. Under these constraints, the total length of the two lines should be minimized. The length of a line is defined as the number of cell borders it passes. In particular, a line connecting cells sharing their border has length 1.

Fig. 6(a) shows an example setting. Fig. 6(b) shows two lines satisfying the constraints above with minimum total length 18.

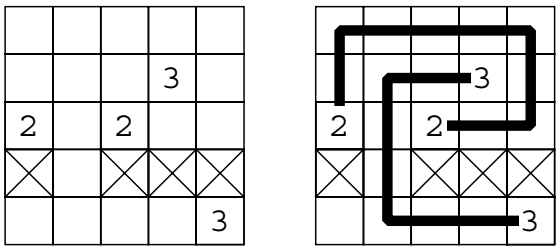

Figure 6: An example setting and its solution

Input

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

$n \quad m$
row_1
$\vdots$
row_n

n is the number of rows which satisfies $2 \leq n \leq 9$. m is the number of columns which satisfies $2 \leq m \leq 9$. Each row_i is a sequence of m digits separated by a space. The digits mean the following.

`0`: Empty

`1`: Occupied by an obstacle

`2`: Marked with "2"

`3`: Marked with "3"

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

Output

For each dataset, one line containing the minimum total length of the two lines should be output. If there is no pair of lines satisfying the requirement, answer "`0`" instead. No other characters should be contained in the output.

Sample Input

```
5 5
0 0 0 0 0
0 0 0 3 0
2 0 2 0 0
1 0 1 1 1
0 0 0 0 3
2 3
2 2 0
0 3 3
6 5
2 0 0 0 0
0 3 0 0 0
0 0 0 0 0
1 1 1 0 0
0 0 0 0 0
0 0 2 3 0
5 9
0 0 0 0 0 0 0 0 0
0 0 0 0 3 0 0 0 0
0 2 0 0 0 0 0 2 0
0 0 0 0 3 0 0 0 0
0 0 0 0 0 0 0 0 0
```

```
9 9
3 0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 3
9 9
0 0 0 1 0 0 0 0 0
0 2 0 1 0 0 0 0 3
0 0 0 1 0 0 0 0 2
0 0 0 1 0 0 0 0 3
0 0 0 1 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
9 9
0 0 0 0 0 0 0 0 0
0 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 0
0 0 0 0 0 0 2 3 2
0 0
```

Output for the Sample Input

```
18
2
17
12
0
52
43
```

Problem N

~~Problem F~~
Power Calculus
Input: F.txt

Starting with x and repeatedly multiplying by x, we can compute x^{31} with thirty multiplications:

$$x^2 = x \times x, \quad x^3 = x^2 \times x, \quad x^4 = x^3 \times x, \quad \ldots, \quad x^{31} = x^{30} \times x.$$

The operation of squaring can appreciably shorten the sequence of multiplications. The following is a way to compute x^{31} with eight multiplications:

$$x^2 = x \times x, \quad x^3 = x^2 \times x, \quad x^6 = x^3 \times x^3, \quad x^7 = x^6 \times x, \quad x^{14} = x^7 \times x^7,$$
$$x^{15} = x^{14} \times x, \quad x^{30} = x^{15} \times x^{15}, \quad x^{31} = x^{30} \times x.$$

This is not the shortest sequence of multiplications to compute x^{31}. There are many ways with only seven multiplications. The following is one of them:

$$x^2 = x \times x, \quad x^4 = x^2 \times x^2, \quad x^8 = x^4 \times x^4, \quad x^{10} = x^8 \times x^2,$$
$$x^{20} = x^{10} \times x^{10}, \quad x^{30} = x^{20} \times x^{10}, \quad x^{31} = x^{30} \times x.$$

There however is no way to compute x^{31} with fewer multiplications. Thus this is one of the most efficient ways to compute x^{31} only by multiplications.

If division is also available, we can find a shorter sequence of operations. It is possible to compute x^{31} with six operations (five multiplications and one division):

$$x^2 = x \times x, \quad x^4 = x^2 \times x^2, \quad x^8 = x^4 \times x^4, \quad x^{16} = x^8 \times x^8, \quad x^{32} = x^{16} \times x^{16},$$
$$x^{31} = x^{32} \div x.$$

This is one of the most efficient ways to compute x^{31} if a division is as fast as a multiplication.

Your mission is to write a program to find the least number of operations to compute x^n by multiplication and division starting with x for the given positive integer n. Products and quotients appearing in the sequence of operations should be x to a positive integer's power. In other words, x^{-3}, for example, should never appear.

Input

The input is a sequence of one or more lines each containing a single integer n. n is positive and less than or equal to 1000. The end of the input is indicated by a zero.

Output

Your program should print the least total number of multiplications and divisions required to compute x^n starting with x for the integer n. The numbers should be written each in a separate line without any superfluous characters such as leading or trailing spaces.

Sample Input

```
1
31
70
91
473
512
811
953
0
```

Output for the Sample Input

```
0
6
8
9
11
9
13
12
```

Problem O

~~Problem G~~
Polygons on the Grid
Input: G.txt

The ultimate Tantra is said to have been kept in the most distinguished temple deep in the sacred forest somewhere in Japan. Paleographers finally identified its location, surprisingly a small temple in Hiyoshi, after years of eager research. The temple has an underground secret room built with huge stones. This underground megalith is suspected to be where the Tantra is enshrined.

The room door is, however, securely locked. Legends tell that the key of the door lock was an integer, that only highest priests knew. As the sect that built the temple decayed down, it is impossible to know the integer now, and the Agency for Cultural Affairs bans breaking up the door. Fortunately, a figure of a number of rods that might be used as a clue to guess that secret number is engraved on the door.

Many distinguished scholars have challenged the riddle, but no one could have ever succeeded in solving it, until recently a brilliant young computer scientist finally deciphered the puzzle. Lengths of the rods are multiples of a certain unit length. He found that, to find the secret number, all the rods should be placed on a grid of the unit length to make one convex polygon. Both ends of each rod must be set on grid points. Elementary mathematics tells that the polygon's area ought to be an integer multiple of the square of the unit length. The area size of the polygon with the largest area is the secret number which is needed to unlock the door.

For example, if you have five rods whose lengths are 1, 2, 5, 5, and 5, respectively, you can make essentially only three kinds of polygons, shown in Figure 7. Then, you know that the maximum area is 19.

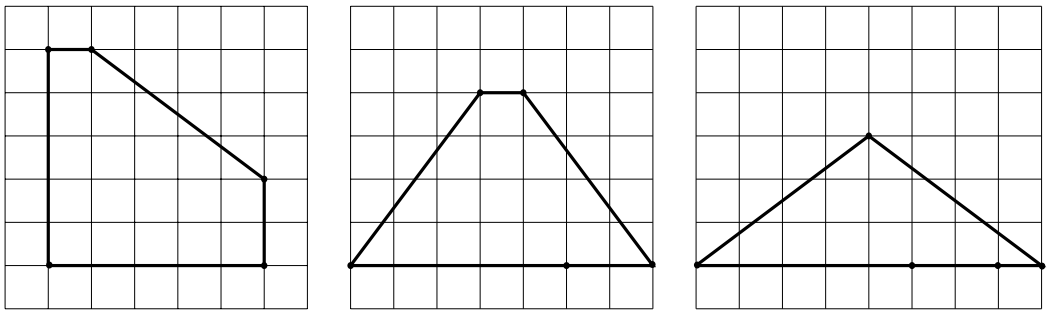

Figure 7: Convex polygons consisting of five rods of lengths 1, 2, 5, 5, and 5

Your task is to write a program to find the maximum area of convex polygons using all the given rods whose ends are on grid points.

Input

The input consists of multiple datasets, followed by a line containing a single zero which indicates the end of the input. The format of a dataset is as follows.

$$
\begin{array}{l}
n \\
r_1 \quad r_2 \quad \cdots \quad r_n
\end{array}
$$

n is an integer which means the number of rods and satisfies $3 \leq n \leq 6$. r_i is an integer which means the length of the i-th rod and satisfies $1 \leq r_i \leq 300$.

Output

For each dataset, output a line containing an integer which is the area of the largest convex polygon. When there are no possible convex polygons for a dataset, output "`-1`".

Sample Input

```
3
3 4 5
5
1 2 5 5 5
6
195 221 255 260 265 290
6
130 145 169 185 195 265
3
1 1 2
6
3 3 3 3 3 3
0
```

Output for the Sample Input

```
6
19
158253
-1
-1
18
```

Problem P

~~Problem H~~

The Best Name for Your Baby
Input: H.txt

In the year 29XX, the government of a small country somewhere on the earth introduced a law restricting first names of the people only to traditional names in their culture, in order to preserve their cultural uniqueness. The linguists of the country specifies a set of rules once every year, and only names conforming to the rules are allowed in that year. In addition, the law also requires each person to use a name of a specific length calculated from one's birth date because otherwise too many people would use the same very popular names. Since the legislation of that law, the common task of the parents of new babies is to find the name that comes first in the alphabetical order among the legitimate names of the given length because names earlier in the alphabetical order have various benefits in their culture.

Legitimate names are the strings consisting of only lowercase letters that can be obtained by repeatedly applying the rule set to the initial string "S", a string consisting only of a single uppercase S.

Applying the rule set to a string is to choose one of the rules and apply it to the string. Each of the rules has the form $A \to \alpha$, where A is an uppercase letter and α is a string of lowercase and/or uppercase letters. Applying such a rule to a string is to replace an occurrence of the letter A in the string to the string α. That is, when the string has the form "$\beta A \gamma$", where β and γ are arbitrary (possibly empty) strings of letters, applying the rule rewrites it into the string "$\beta \alpha \gamma$". If there are two or more occurrences of A in the original string, an arbitrary one of them can be chosen for the replacement.

Below is an example set of rules.

S	$\to$	aAB	(1)
A	$\to$		(2)
A	$\to$	Aa	(3)
B	$\to$	AbbA	(4)

Applying the rule (1) to "S", "aAB" is obtained. Applying (2) to it results in "aB", as A is replaced by an empty string. Then, the rule (4) can be used to make it "aAbbA". Applying (3) to the first occurrence of A makes it "aAabbA". Applying the rule (2) to the A at the end results in "aAabb". Finally, applying the rule (2) again to the remaining A results in "aabb". As no uppercase letter remains in this string, "aabb" is a legitimate name.

We denote such a rewriting process as follows.

$$\mathrm{S} \xrightarrow{(1)} \mathrm{aAB} \xrightarrow{(2)} \mathrm{aB} \xrightarrow{(4)} \mathrm{aAbbA} \xrightarrow{(3)} \mathrm{aAabbA} \xrightarrow{(2)} \mathrm{aAabb} \xrightarrow{(2)} \mathrm{aabb}$$

Linguists of the country may sometimes define a ridiculous rule set such as follows.

$$\begin{array}{lcll} \mathrm{S} & \rightarrow & \mathrm{sA} & (1) \\ \mathrm{A} & \rightarrow & \mathrm{aS} & (2) \\ \mathrm{B} & \rightarrow & \mathrm{b} & (3) \end{array}$$

The only possible rewriting sequence with this rule set is:

$$\mathrm{S} \xrightarrow{(1)} \mathrm{sA} \xrightarrow{(2)} \mathrm{saS} \xrightarrow{(1)} \mathrm{sasA} \xrightarrow{(2)} \cdots$$

which will never terminate. No legitimate names exist in this case. Also, the rule (3) can never be used, as its left hand side, B, does not appear anywhere else.

It may happen that no rules are supplied for some uppercase letters appearing in the rewriting steps. In its extreme case, even S might have no rules for it in the set, in which case there are no legitimate names, of course. Poor nameless babies, sigh!

Now your job is to write a program that finds the name earliest in the alphabetical order among the legitimate names of the given length conforming to the given set of rules.

Input

The input is a sequence of datasets, followed by a line containing two zeros separated by a space representing the end of the input. Each dataset starts with a line including two integers n and l separated by a space, where n ($1 \leq n \leq 50$) is the number of rules and l ($0 \leq l \leq 20$) is the required length of the name. After that line, n lines each representing a rule follow. Each of these lines starts with one of uppercase letters, A to Z, followed by the character "=" (instead of "$\rightarrow$") and then followed by the right hand side of the rule which is a string of letters A to Z and a to z. The length of the string does not exceed 10 and may be zero. There appears no space in the lines representing the rules.

Output

The output consists of the lines showing the answer to each dataset in the same order as the input. Each line is a string of lowercase letters, a to z, which is the first legitimate name conforming to the rules and the length given in the corresponding input dataset. When the given set of rules has no conforming string of the given length, the corresponding line in the output should show a single hyphen, "-". No other characters should be included in the output.

Sample Input

```
4 3
A=a
A=
S=ASb
S=Ab
2 5
S=aSb
S=
1 5
S=S
1 0
S=S
1 0
A=
2 0
A=
S=AA
4 5
A=aB
A=b
B=SA
S=A
4 20
S=AAAAAAAAAA
A=aA
A=bA
A=
0 0
```

Output for the Sample Input

```
abb
-
-
-
-

aabbb
aaaaaaaaaaaaaaaaaaaa
```

Problem Q

~~Problem I~~
Enjoyable Commutation
Input: I.txt

Isaac is tired of his daily trip to his office, using the same shortest route everyday. Although this saves his time, he must see the same scenery again and again. He cannot stand such a boring commutation any more.

One day, he decided to improve the situation. He would change his route everyday at least slightly. His new scheme is as follows. On the first day, he uses the shortest route. On the second day, he uses the second shortest route, namely the shortest except one used on the first day. In general, on the k-th day, the k-th shortest route is chosen. Visiting the same place twice on a route should be avoided, of course.

You are invited to help Isaac, by writing a program which finds his route on the k-th day. The problem is easily modeled using terms in the graph theory. Your program should find the k-th shortest path in the given directed graph.

Input

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

$$
\begin{array}{llllll}
n & m & k & a & b \\
x_1 & y_1 & d_1 \\
x_2 & y_2 & d_2 \\
 & \vdots \\
x_m & y_m & d_m
\end{array}
$$

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

n is the number of nodes in the graph. You can assume the inequality $2 \leq n \leq 50$. m is the number of (directed) edges. a is the start node, and b is the goal node. They are between 1 and n, inclusive. You are required to find the k-th shortest path from a to b. You can assume $1 \leq k \leq 200$ and $a \neq b$.

The i-th edge is from the node x_i to y_i with the length d_i ($1 \leq i \leq m$). Both x_i and y_i are between 1 and n, inclusive. d_i is between 1 and 10000, inclusive. You can directly go from x_i to y_i, but not from y_i to x_i unless an edge from y_i to x_i is explicitly given. The edge connecting the same pair of nodes is unique, if any, that is, if $i \neq j$, it is never the case that x_i equals x_j

and y_i equals y_j. Edges are not connecting a node to itself, that is, x_i never equals y_i. Thus the inequality $0 \leq m \leq n(n-1)$ holds.

Note that the given graph may be quite unrealistic as a road network. Both the cases $m = 0$ and $m = n(n-1)$ are included in the judges' data.

The last dataset is followed by a line containing five zeros (separated by a space).

Output

For each dataset in the input, one line should be output as specified below. An output line should not contain extra characters such as spaces.

If the number of distinct paths from a to b is less than k, the string `None` should be printed. Note that the first letter of `None` is in uppercase, while the other letters are in lowercase.

If the number of distinct paths from a to b is k or more, the node numbers visited in the k-th shortest path should be printed in the visited order, separated by a hyphen (minus sign). Note that a must be the first, and b must be the last in the printed line.

In this problem the term *shorter* (thus *shortest* also) has a special meaning. A path P is defined to be shorter than Q, if and only if one of the following conditions holds.

1. The length of P is less than the length of Q. The length of a path is defined to be the sum of lengths of edges on the path.
2. The length of P is equal to the length of Q, and P's sequence of node numbers comes earlier than Q's in the dictionary order. Let's specify the latter condition more precisely. Denote P's sequence of node numbers by $p_1, p_2, \ldots, p_s$, and Q's by $q_1, q_2, \ldots, q_t$. $p_1 = q_1 = a$ and $p_s = q_t = b$ should be observed. The sequence P comes earlier than Q in the dictionary order, if for some r ($1 \leq r \leq s$ and $r \leq t$), $p_1 = q_1, \ldots, p_{r-1} = q_{r-1}$, and $p_r < q_r$ (p_r is numerically smaller than q_r).

A path visiting the same node twice or more is not allowed.

Sample Input

```
5 20 10 1 5
1 2 1
1 3 2
1 4 1
1 5 3
2 1 1
2 3 1
2 4 2
2 5 2
```

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

Output for the Sample Input

```
1-2-4-3-5
1-2-3-4
None
```

In the case of the first dataset, there are 16 paths from the node 1 to 5. They are ordered as follows (The number in parentheses is the length of the path).

1	(3)	1-2-3-5	9	(5)	1-2-3-4-5
2	(3)	1-2-5	10	(5)	1-2-4-3-5
3	(3)	1-3-5	11	(5)	1-2-4-5
4	(3)	1-4-3-5	12	(5)	1-3-4-5
5	(3)	1-4-5	13	(6)	1-3-2-5
6	(3)	1-5	14	(6)	1-3-4-2-5
7	(4)	1-4-2-3-5	15	(6)	1-4-3-2-5
8	(4)	1-4-2-5	16	(8)	1-3-2-4-5