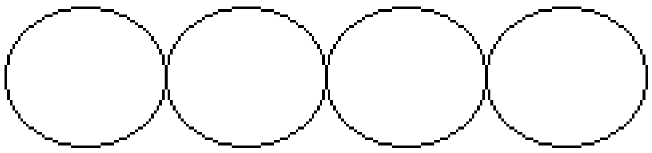
UVa Online Judge

11001 Necklace

The people of a certain tribe produce circular ceramic discs with equal diameter by some rare clay. A necklace is formed by connecting one or more discs. The figure below shows a necklace made with 4 discs. Its length is 4 times the diameter of each disc.



The thickness of each disc is fixed. The diameter D and the volume of clay used V has the following relationship:

$$p(x) = \begin{cases} 0.3\sqrt{V - V_0} & \text{if} \quad V > V_0 \\ 0 & \text{if} \quad V \le V_0 \end{cases}$$

where V_0 is the volume consumed in the baking process, in the same unit of V. When $V \le V_0$, no ceramic discs can be made. As an example, let $V_{total} = 10$, $V_0 = 1$. If we use it to make 1 disc, $V = V_{total} = 10$, $D = 0.9$. If we divide the clay into 2 parts, the volume of each part $V = V_{total}/2 = 5$, and diameter of each disc formed is $D' = 0.3\sqrt{5 - 1} = 0.6$, thus the length of necklace formed this way is 1.2.

As per the above example, it is obvious that the lengths of necklaces differ as the number of discs made changes. Please write a program that computes the number of discs one should make so that the necklace formed is the longest.

Input

Each line of input contains two numbers, V_{total} $(0 < V_{total} \le 60000)$ and V_0 $(0 < V_0 \le 600)$, as defined above. Input ends with a case where $V_{total} = V_0 = 0$.

Output

Each line of output should give the number of discs one should make so that the necklace formed is the longest. If this number is not unique, or no necklaces can be formed at all, output '0' instead.

Sample Input

```
10 1
10 2
0 0
```

Sample Output

```
5
0
```

12498 Ant's Shopping Mall

In the world of ant there is a popular shopping mall named Ants Shopping Mall. The shopping mall is a grid with R rows and C columns. Any cell of the grid can be occupied by a shop or empty. The shopping mall is dynamic in a sense that if there is a shop at cell (r, c) (where $1 \leq r \leq R$ and $1 \leq c \leq C$, here r defines the row and c defines the column of the cell) then in a single move the shop can be moved to $(r, c-1)$ or $(r, c+1)$ if the position is empty. But it cannot be moved outside the grid.

The ant queen now wants to visit the shopping mall. The queen can move vertically that is if the queen is at cell (r, c) then in the next step she can be at $(r+1, c)$. She has a special way of visit, she always starts from first row that is from any cell $(1, c)$ and continues until the last row is reached that is cell (R, c). The owner of the shopping mall wants to impress queen so he wants to find a path of the form $(1, c)$, $(2, c)$, ..., (R, c) for the queen in advance such that there is no shop in any cell of the path. For this the owner of the mall may need to move zero or more shops but he wants to do it in minimum number of shop movement. Now the owner hired you to solve the task for him. If it is not possible to find a path then you have to report it also.

Input

First line of the input contains a positive number T, number of test cases. There will be at most 50 test cases. For each test case the first line contains R and C separated by spaces ($2 \leq R \leq 50$ and $1 \leq C \leq 50$). Each of the next R lines contains C characters, each of them is either '`0`' or '`1`'. If the j-th character in the i-th line contains '`1`' then cell (i, j) contains a shop otherwise cell (i, j) is empty.

Output

For each test you have to output minimum number of moves needed if it is possible to generate a path for the queen. Otherwise output '`-1`'. See output format for clarification.

Sample Input

```
3
2 4
1010
0101
3 3
111
111
111
3 5
01111
11110
11011
```

Sample Output

```
Case 1: 1
Case 2: -1
Case 3: 4
```

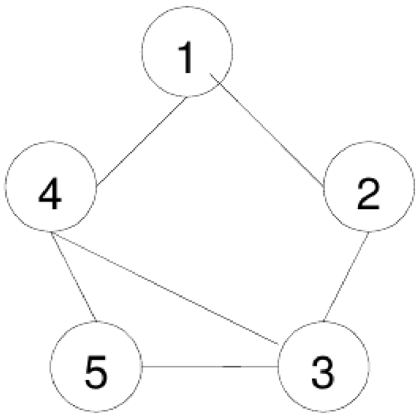
677 All Walks of length n from the first node

A computer network can be represented as a graph. Let G = (V, E) be an undirected graph, V = $(v_1, v_2, v_3, \ldots, v_m)$ represents all nodes, where m is the number of nodes, and E represents all edges. The first node is v_1 and the last node is v_m . The number of edges is k. Define the adjacency matrix $A = (a_{ij})_{m \times m}$ where

$$a_{ij} = 1 \text{ if } \{v_i, v_j\} \in \text{E, otherwise } a_{ij} = 0$$

An example of the adjacency matrix and its corresponding graph are as follows:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$



Calculate

$$A^n = \underbrace{A \cdot A \cdots A}_{n}$$

and use the Boolean operations where $0 + 0 = 0, 0 + 1 = 1 + 0 = 1, 1 + 1 = 1$, and $0 \bullet 0 = 0 \bullet 1 = 1 \bullet 0 = 0, 1 \bullet 1 = 1$. The entry in row i and column j of A^n is 1 if and only if at least there exists a walk of length n between the i-th and j-th nodes of V. In other words, the distinct walks of length n between the i-th and j-th nodes of V may be more than one. Note that the node in the paths can be repetitive.

The following example shows the walks of length 2.

$$A^2 = A \cdot A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Write programs to do above calculation and print out all distinct walks of length n. (In this problem we let the maximum walks of length n be 5 and the maximum number of nodes be 10.)

Input

The input file contains a number of test examples, the test examples are separated by '`-9999`'. Each test example consists of the number of nodes and the length of walks in the first row, and then the adjacency matrix.

Output

The output file must contain all distinct walks of the length n, and with all its nodes different, from the first node, listed in lexicographical order. In case there are not walks of length n, just print '`no walk of length` n'

Separate the output of the different cases by a blank line.

Sample Input

```
5 2
0 1 0 1 0
1 0 1 0 0
0 1 0 1 1
1 0 1 0 1
0 0 1 1 0
-9999
5 3
0 1 0 1 0
1 0 1 0 0
0 1 0 1 1
1 0 1 0 1
0 0 1 1 0
```

Sample Output

```
(1,2,3)
(1,4,3)
(1,4,5)

(1,2,3,4)
(1,2,3,5)
(1,4,3,2)
(1,4,3,5)
(1,4,5,3)
```

12515 Movie Police

Movie Police (MP) is an international top secret law enforcement agency, controlling illegal movie downloads on internet. With their elite team of programmers, MP has developed a very smart algorithm to produce movie signatures. A *movie signature* is a binary string, one bit for every frame in the movie, so that the i-th bit in the signature corresponds to the i-th frame in the movie. The algorithm is so amazing that it outputs consistently the same signature for versions of the same film with different quality resolutions. One application of this revolutionary technology uses it to detect if a small clip is part of a movie, looking for a high similarity between the clip signature and the movie signature.

Now MP has started to apply this technology and, as a first step, a massive online database of movie signatures was already built. As a new member of the MP crew, you must write a program that, given the signature of a clip, finds the index in the MP database of a movie whose signature *matches* the clip signature at most. That is, a movie whose signature has a substring, of the same length of the clip signature, that is most *similar* to the clip signature.

Similarity between strings of the same length is defined by means of their *Hamming distance* (number of bits that do not match), so that "more similar" means "less Hamming distance".

Input

The first line of the input contains two positive integer numbers M and Q, separated by a blank, where M indicates the number of movie signatures in the database and Q indicates the number of clip signatures to process ($1 \leq M \leq 1000$, $1 \leq Q \leq 500$). Each one of the following M lines contains a binary string s_i describing the i-th movie signature in the database. You may suppose that s_i has length l_i, where $1 \leq l_i \leq 100$. Finally, there are Q lines, each one with a binary string that corresponds to a clip signature to search for maximal similarity in the database. You may assume that, for every clip signature to be searched, there is at least one movie signature in the database whose length is greater or equal than the clip's length.

Output

For each clip signature given in the input, output a single line with the lowest index i of a movie s_i ($1 \leq i \leq M$) that matches the clip at most, as above explained. If there are two movie signatures that match the clip signature maximally, answer the one with lower index in the database.

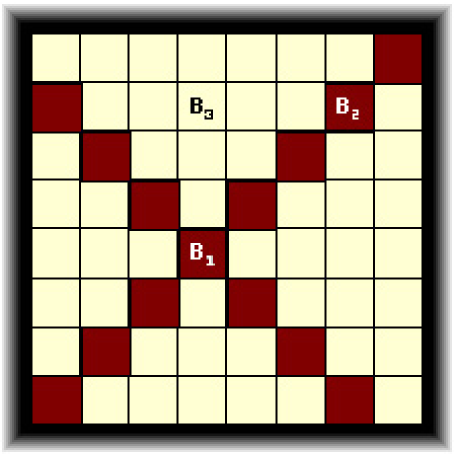
Sample Input

```
3 1
000011
1101111111
1111100000
1000111
```

Sample Output

```
2
```

861 Little Bishops



A bishop is a piece used in the game of chess which can only move diagonally from its current position. Two bishops attack each other if one is on the path of the other. In the figure below, the dark squares represent the reachable locations for bishop B_1 from its current position. Bishops B_1 and B_2 are in attacking position, while B_1 and B_3 are not. Bishops B_2 and B_3 are also in non-attacking position.

Given two numbers n and k, determine the number of ways one can put k bishops on an $n \times n$ chessboard so that no two of them are in attacking positions.

Input

The input file may contain multiple test cases. Each test case occupies a single line in the input file and contains two integers n $(1 \leq n \leq 8)$ and k $(0 \leq k \leq n^2)$.

A test case containing two zeros terminates the input.

Output

For each test case, print a line containing the total number of ways one can put the given number of bishops on a chessboard of the given size so that no two of them lie in attacking positions. You may safely assume that this number will be less than 10^{15}.

Sample Input

```
8 6
4 4
0 0
```

Sample Output

```
5599888
260
```

927 Integer Sequences from Addition of Terms

We consider sequences formed from the addition of terms of a given sequence. Let $\{a_n\}$, $n = 1, 2, 3, \ldots$, be an arbitrary sequence of integer numbers; d a positive integer. We construct another sequence $\{b_m\}$, $m = 1, 2, 3, \ldots$, by defining b_m as consisting of $n \times d$ occurrences of the term a_n:

$$b_1 = \underbrace{a_1, \ldots, a_1}_{d \text{ occurrences of } a_1}, b_2 = \underbrace{a_2, \ldots, a_2}_{2d \text{ occurrences of } a_2}, b_3 = \underbrace{a_3, \ldots, a_3}_{3d \text{ occurrences of } a_3}, \cdots$$

For example, if $a_n = n$, and $d = 1$,then the resulting sequence $\{b_m\}$ is:

$$\underbrace{1}_{b_1}, \underbrace{2, 2}_{b_2}, \underbrace{3, 3, 3}_{b_3}, \underbrace{4, 4, 4, 4}_{b_4}, \cdots$$

Given a_n and d we want to obtain the corresponding k-th integer in the sequence $\{b_m\}$. For example, with $a_n = n$ and $d = 1$ we have 3 for $k = 6$; we have 4 for $k = 7$. With $a_n = n$ and $d = 2$, we have 2 for $k = 6$; we have 3 for $k = 7$.

Input

The first line of input contains C $(0 < C < 100)$, the number of test cases that follows.

Each of the C test cases consists of three lines:

1. The first line represents a_n — a polynomial in n of degree i with non-negative integer coefficients in increasing order of the power:

$$a_n = c_0 + c_1 n + c_2 n^2 + c_3 n^3 + \cdots + c_i n^i$$

   where $c_j \in \mathbb{N}_0$, $j = 0, \ldots, i$. This polynomial a_n is codified by its degree i followed by the coefficients c_j, $j = 0, \ldots, i$. All the numbers are separated by a single space.

2. The second line is the positive integer d.

3. The third line is the positive integer k.

It is assumed that the polynomial a_n is a polynomial of degree less or equal than 20 $(1 \leq i \leq 20)$ with non-negative integer coefficients less or equal than 10000 $(0 \leq c_j \leq 10000$, $j = 0, \ldots, i)$; $1 \leq d \leq 100000$; $1 \leq k \leq 1000000$.

Output

The output is a sequence of lines, one for each test case. Each of these lines contains the k-th integer in the sequence $\{b_m\}$ for the corresponding test case. This value is less or equal than $2^{63} - 1$.

Sample Input

```
2
4 3 0 0 0 23
25
100
1 0 1
1
6
```

Sample Output

```
1866
3
```