A

Assembly line

The last worker in a production line at the factory of Automated Composed Machinery is worried. She knows that her job hangs in the balance unless her productivity increases. Her work consists of assembling a set of pieces in a given sequence, but the time spent on assembling pieces a and b and then c may not the same as that on assembling pieces b and c, and then assembling a with the resulting component. Only two consecutive pieces may be assembled at a time, and once they are assembled they behave as another piece in terms of the time needed for further assembly.

In order to aid her, you need to find the optimal way to assemble all components. The input to your program will be a set of symbols representing (types of) pieces, and a so-called assembly table representing the time it takes to assemble them, as well as the type of the resulting component. For instance, we may have two symbols $\{a, b\}$, and the following table:

	a	b
a	3-b	5-b
b	6-a	2-b

This means, for example, that two pieces of type a and a may assembled in 3 minutes, and the result is a component of type b, in that the time required to assemble it again with another piece of, say, type a is 6 minutes, and so on. Note that the table is not symmetric, i.e. assembling b and a may be more time-consuming than a and b.

For a sequence of components labelled aba, the two possible solutions are:

- $(ab)a = ba = a$ with time $time(ab) + time(ba) = 5 + 6 = 11$.
- $a(ba) = aa = b$ with time $time(ba) + time(aa) = 6 + 3 = 9$.

So the result for this case would be a piece of type b in 9 minutes (denoted 9-b).

Input

The input consists of several test cases. Each test case begins with a line containing a natural number k ($1 \leq k \leq 26$), followed by a line with k symbols (characters in `[a-z]`) separated by spaces. The following k lines contain the assembly table: the i-th line has k pairs of the form *time*-*result*, where *time* is an integer between 0 and 1 000 000 inclusive, and *result* a symbol belonging to the preceding set. The j-th pair in the i-th line represents the time to compose pieces of types represented by the i-th and j-th symbols, along with the type of the resulting piece. After the table, a line with an integer n indicates the number of lines that follow, each line being a string of at most 200 symbols. Each of these lines is a sequence of components that need to be assembled together in the right order.

The input will finish with a line containing `0`, which should not be processed.

Output

For each test case, print n lines, each with an integer *time* and a symbol *result* in the format *time*-*result*. Each line represents the minimum time and the type of the resulting piece for the corresponding case in the input. In case of a tie among several possible results with the same minimum time, choose from among those the piece whose type letter appears first in the line that contained the k symbols at the beginning of the test case. (For example, if that line was $a\ c\ b$ and both c and b can be obtained with minimum cost 5, print 5-c).

There must be an empty line between the output of different test cases.

Sample Input

```
2
a b
3-b 5-b
6-a 2-b
2
aba
bba
2
m e
5-e 4-m
3-e 4-m
1
eme
0
```

Sample Output

```
9-b
8-a

7-m
```

B

Stammering Aliens

Dr. Ellie Arroway has established contact with an extraterrestrial civilization. However, all efforts to decode their messages have failed so far because, as luck would have it, they have stumbled upon a race of stuttering aliens! Her team has found out that, in every long enough message, the most important words appear repeated a certain number of times as a sequence of consecutive characters, even in the middle of other words. Furthermore, sometimes they use contractions in an obscure manner. For example, if they need to say *bab* twice, they might just send the message *babab*, which has been abbreviated because the second *b* of the first word can be reused as the first *b* of the second one.

Thus, the message contains possibly overlapping repetitions of the same words over and over again. As a result, Ellie turns to you, S.R. Hadden, for help in identifying the gist of the message.

Given an integer m, and a string s, representing the message, your task is to find the longest substring of s that appears at least m times. For example, in the message *baaaababababbababbab*, the length-5 word *babab* is contained 3 times, namely at positions 5, 7 and 12 (where indices start at zero). No substring appearing 3 or more times is longer (see the first example from the sample input). On the other hand, no substring appears 11 times or more (see example 2).

In case there are several solutions, the substring with the rightmost occurrence is preferred (see example 3).

Input

The input contains several test cases. Each test case consists of a line with an integer m ($m \geq 1$), the minimum number of repetitions, followed by a line containing a string s of length between m and 40 000, inclusive. All characters in s are lowercase characters from "a" to "z". The last test case is denoted by $m = 0$ and must not be processed.

Output

Print one line of output for each test case. If there is no solution, output `none`; otherwise, print two integers in a line, separated by a space. The first integer denotes the maximum length of a substring appearing at least m times; the second integer gives the rightmost starting position of this substring.

Sample Input

```
3
baaaababababbababbab
11
baaaababababbababbab
3
cccccc
0
```

Sample Output

```
5 12
none
4 2
```

– LCM Pair Sum –

One of your friends desperately needs your help. He is working with a secret agency and doing some encoding stuffs. As the mission is confidential he does not tell you much about that, he just want you to help him with a special property of a number. This property can be expressed as a function $f(n)$ for a positive integer n. It is defined as:

$$f(n) = \sum_{\substack{1 \le p \le q \le n \\ lcm(p,q)=n}} (p+q)$$

In other words, he needs the sum of all possible pairs whose least common multiple is n. (The least common multiple (LCM) of two numbers p and q is the lowest positive integer which can be perfectly divided by both p and q). For example, there are 5 different pairs having their LCM equal to 6 as (1, 6), (2, 6), (2, 3), (3, 6), (6, 6). So $f(6)$ is calculated as $f(6) = (1+6)+(2+6)+(2+3)+(3+6)+(6+6) = 7+8+5+9+12 = 41$.

Your friend knows you are good at solving this kind of problems, so he asked you to lend a hand. He also does not want to disturb you much, so to assist you he has factorized the number. He thinks it may help you.

INPUT

The first line of input will contain the number of test cases T ($T \le 500$). After that there will be T test cases. Each of the test cases will start with a positive number C ($C \le 15$) denoting the number of prime factors of n. Then there will be C lines each containing two numbers P_i and a_i denoting the prime factor and its power (P_i is a prime between 2 and 1000) and ($1 \le a_i \le 50$). All the primes for an input case will be distinct.

OUTPUT

For each of the test cases produce one line of output denoting the case number and $f(n)$ modulo 1000000007. See the output for sample input for exact formatting.

INPUT EXAMPLE

```
3
2
2 1
3 1
2
2 2
3 1
1
5 1
```

OUTPUT EXAMPLE

```
Case 1: 41
Case 2: 117
Case 3: 16
```

– Countdown –

The "Countdown" TV show has a part that consists of obtaining a number by combining six different numbers using the basic mathematical operations: addition, subtraction, product and division. The basic rules for the game are:

- The contestant selects six of twenty-four shuffled tiles. The tiles are arranged into two groups: four "large numbers" (25, 50, 75 and 100) and the remainder "small numbers", which comprise two each of the numbers 1 to 10. Hence the tiles have the values {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100}.
- The contestant chooses how many large numbers are in the selection; anywhere from none.
- The contestants then have thirty seconds to get a number as close to the target as possible by combining the six selected numbers using addition, subtraction, multiplication and division.
- Not all numbers need to be used.
- A number can be used as many times as it appears.
- Fractions are not allowed, only positive integers may be used at any stage of the calculation.

Example:

- Contestant requests two large numbers and four small numbers.
- Selection is: 75 50 2 3 8 7
- Randomly generated target is: 812
- Contestant declares result: 813
- Contestant gives details: $75 + 50 = 125$; 125 - 8 = 117; $117 \times 7 = 819$; $3 \times 2 = 6$; 819 - 6 = 813
- Expert notes: $50 + 8 = 58$; $7 \times 2 = 14$; $14 \times 58 = 812$

Your task is to write a program that calculates the best sequence of operations that lead to the target number T. If there is no way to get T, give the closest solution.

INPUT

The input consists of several cases, one per line. The first line indicates the number of cases C to process ($1 \leq C \leq 50$). Each of the following C lines contains six natural numbers from the set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100} and another natural number T ($1 \leq T \leq 999$) that indicates the target.

OUTPUT

The output for each case will be a set of lines with the following format:

- First line: `Target:` number T
- n lines: sequence of operations, the format is $operand_2$ $operator$ $operand_2$ $=$ $result$
- Last line: `Best approx:` number obtained
- Blank line

See example for a better understanding. If there is more than one best approximation, all of them will be considered valid. The sequence of operations should be valid, you should never use a value before you obtain it. It is OK to print more operations than needed as long as they are valid. Note that all the numbers and operators must be separated by at least one space.

INPUT EXAMPLE

```
3
1 75 100 5 3 25 25
100 100 100 100 100 75 345
1 3 1 10 100 75 345
```

OUTPUT EXAMPLE

```
Target: 25
Best approx: 25

Target: 345
100 + 100 = 200
75 + 100 = 175
200 * 175 = 35000
35000 - 100 = 34900
34900 / 100 = 349
Best approx: 349

Target: 345
100 - 10 = 90
3 * 90 = 270
270 + 75 = 345
Best approx: 345
```

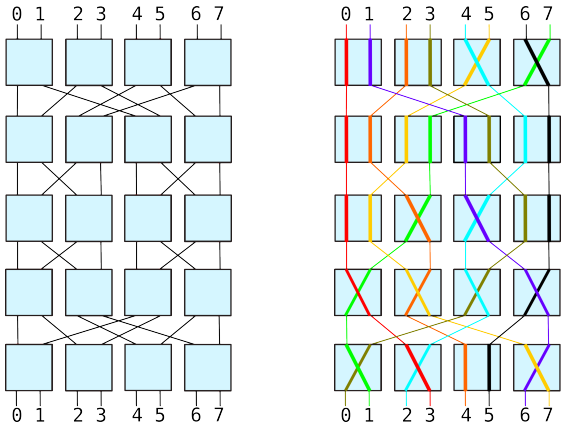
E

Routing

You work as an engineer for the *Inane Collaboration for Performance Computing*, where you are in charge of designing an intercommunication network for their computers. The network is arranged as a rectangular array of $2n-1$ rows, each having 2^{n-1} switches. A switch is a device with two input wires, X and Y, and two output wires, X' and Y'. If the switch is off, data from input X will be relayed to output X', and data from Y to Y'. If it is on, X will be connected to Y' and Y to X'. Additionally, there are 2^n computers in the topmost and bottommost rows, and messages need to be sent between pairs of them. Notice that data from two different sources cannot share a wire but, of course, both pieces of data can be routed through the same switch on different inputs.

You have come to the conclusion that the network that best suits your purposes has the Beneš topology. A 1-Beneš network is just a switch. For $n > 1$, a n-Beneš network can be constructed recursively as follows:

- In the first (top) row there are 2^{n-1} switches such that switch j ($0 \leq j < 2^{n-1}$) has data inputs from computers $2j$ and $2j+1$ (we label the computers in the topmost and bottommost rows with integers between 0 and $2^n - 1$, inclusive, from left to right).
- Then a *perfect shuffle* permutation is applied to the output wires between the first and the second rows of switches, meaning that output number j in a row is connected to input number j' in the next row, where j' is obtained by rotating the n-bit pattern representing j in binary one bit to the right (again, inputs and outputs are numbered from left to right).
- If $n > 2$, the next rows of switches, up to (and including) the last-but-one, form two $(n-1)$-Beneš subnetworks, one on the left side and the other on the right side.
- Finally, the *inverse* shuffle permutation is applied to the outputs and a last row of switches is added.



(a) Topology

(b) Switch states for second sample input

Figure 4: 3-Benes network

For example, Figure **??** shows the Beneš network for $n = 3$ (squares represent switches; computers in the top and bottom rows are not drawn, but assigned with integers from 0 to 7). Figure **??** shows a possible state of the switches; squares where two of the lines cross are switches that have been turned on. You may verify that this state allows us to simultaneously establish communication paths from computers $0, 1, 2, 3, 4, 5, 6, 7$ at the bottom to $3, 7, 4, 0, 2, 6, 1, 5$ at the top, respectively.

You are given a set of pairs (a, b) of computers to connect simultaneously (where a is a computer in the bottom row and b a computer in the top row) by means of wire-disjoint paths, and you are to find how to select the state of all switches so that this can be accomplished.

Input

The first line of each test case is an integer n ($1 \leq n \leq 13$), meaning that you have 2^n pairs of computers to connect, as described above. A line with $n = 0$ marks the end of the input and should not be processed.

Each line with $n > 0$ will be followed by another line containing 2^n integers. The i-th integer ($0 \leq i < 2^n$) will be the computer in the topmost row that the i-th computer in the bottommost row needs to communicate with.

Output

The output for each case should have $2n-1$ lines, each containing a binary string of length 2^{n-1} indicating, for each switch, whether it must be turned on (1) or off (0).

The input given will always have at least one solution. In case of several solutions, return the lexicographically smallest one. That is, the string in the top row must be lexicographically smallest; in case of a tie, the string in the second row must be lexicographically smallest, and so on.

Outputs for different test cases should be separated by a blank line.

Sample Input

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

Sample Output

```
00
11
11

0011
0000
0110
1111
1101
```

F

Almost Shortest Path

Source file name: `almost.c`, `almost.cpp` *or* `almost.java`

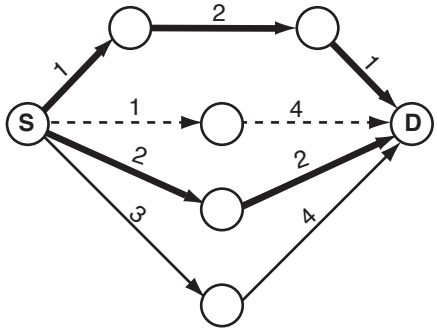
Finding the shortest path that goes from a starting point to a destination point given a set of points and route lengths connecting them is an already well known problem, and it's even part of our daily lives, as shortest path programs are widely available nowadays.

Most people usually like very much these applications as they make their lives easier. Well, maybe not that much easier.

Now that almost everyone can have access to GPS navigation devices able to calculate shortest paths, most routes that form the shortest path are getting slower because of heavy traffic. As most people try to follow the same path, it's not worth it anymore to follow these directions.

With this in his mind, your boss asks you to develop a new application that only he will have access to, thus saving him time whenever he has a meeting or any urgent event. He asks you that the program must answer not the shortest path, but the almost shortest path. He defines the almost shortest path as the shortest path that goes from a starting point to a destination point such that no route between two consecutive points belongs to any shortest path from the starting point to the destination.

For example, suppose the figure below represents the map given, with circles representing location points, and lines representing direct, one-way routes with lengths indicated. The starting point is marked as `S` and the destination point is marked as `D`. The bold lines belong to a shortest path (in this case there are two shortest paths, each with total length 4). Thus, the almost shortest path would be the one indicated by dashed lines (total length 5), as no route between two consecutive points belongs to any shortest path. Notice that there could exist more than one possible answer, for instance if the route with length 3 had length 1. There could exist no possible answer as well.



Input

The input contains several test cases. The first line of a test case contains two integers N ($2 \leq N \leq 500$) and M ($1 \leq M \leq 10^4$), separated by a single space, indicating respectively the number of points in the map and the number of existing one-way routes connecting two points directly. Each point is identified by an integer between 0 and $N-1$. The second line contains two integers S and D, separated by a single space, indicating respectively the starting and the destination points ($S \neq D$; $0 \leq S, D < N$). Each one of the following M lines contains three integers U, V and P ($U \neq V$; $0 \leq U, V < N$; $1 \leq P \leq 10^3$), separated by single spaces, indicating the existence of a one-way route from U to V with distance P. There is at most one

route from a given point U to a given point V, but notice that the existence of a route from U to V does not imply there is a route from V to U, and, if such road exists, it can have a different length. The end of input is indicated by a line containing only two zeros separated by a single space.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing `-1` if it is not possible to match the requirements, or an integer representing the length of the almost shortest path found.

The output must be written to standard output.

Sample input	**Output for the sample input**
7 9 0 6 0 1 1 0 2 1 0 3 2 0 4 3 1 5 2 2 6 4 3 6 2 4 6 4 5 6 1 4 6 0 2 0 1 1 1 2 1 1 3 1 3 2 1 2 0 3 3 0 2 6 8 0 1 0 1 1 0 2 2 0 3 3 2 5 3 3 4 2 4 1 1 5 1 1 3 0 1 0 0	5 -1 6