2012 National Collegiate Programming Contest

- Problems: There are 10 problems (20 pages in all, not counting this cover page) in this packet.
- Problem Input: Input to the problems are through the input files. Input filenames are given in the table below. Each input file may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Problem Output: All output should be directed to standard output (screen output).
- Time Limit: The judges will run each submitted program with certain time limit (given in the table below).

Table 1: Problem Information Sheet

	Problem Name	Input File	Time Limit
Problem A	String Editor	pa.in	3 secs.
Problem B	Integer Partition	pb.in	5 secs.
Problem C	Matrix	pc.in	5 secs.
Problem D	A Matrix Decipher	pd.in	3 secs.
Problem E	Hostage Rescue	pe.in	15 secs.
Problem F	Optimal Transformation Cost	pf.in	3 secs.
Problem G	SHA-4	pg.in	3 secs.
Problem H	Gap Verification	ph.in	5 secs.
Problem I	Christmas Gifts	pi.in	3 secs.
Problem J	The Resequencing LCS Problem	pj.in	15 secs.

Problem A
String Editor

Input File: *pa.in*
Time Limit: *3 seconds*

Prof. Chiu teaches English at Not-Simplifying-Your-Sentence University (NSYS U. for short). Everyday he needs to correct the sentences written by students. In order to let the students know their mistakes, Prof. Chiu always tells students how to revise the sentences instead of only giving them the correct ones. To this aim, he gives a series of editing commands for a wrong sentence. Most of his students can realize the editing commands and obtain the correct revision. But misunderstandings somehow happen occasionally.

To help the students of NSYN U., you are asked to design a string editing program. For a given sentence and a series of editing commands, the program can print the resulting sentence. To make the commands clear, we use a cursor and all the commands are executed at the cursor position. The following editing commands should be supported, in which c denotes the current position of the cursor, n is a positive integer, and s is a string. Initially the cursor is at the first (left-most) position ($c = 0$).

Table 2: Editing Commands

Command	Operations
<<	Backspace, i.e., delete the character left to the cursor. If $c = 0$, then do nothing.
<[n]	Shift cursor left by n characters. If $c < n$, c is set to be 0.
>[n]	Shift cursor right by n characters. Blanks will be appended if necessary.
ˆ[s]	Insert s at the cursor position and the cursor position will be moved to the end of s.
#[s]	Replace with s at the cursor position and the cursor position will be moved to the end of s.
![n]	Delete n characters from the cursor position.

Technical Specification

1. The editing sentence, i.e., the sentence to be edited according to the commands, is a string consisting of only alphanumeric characters and spaces.
2. The length of the editing sentence is at most 100.
3. The length of command string is at most 200.

Input File Format

The input file contains several test cases. For each test case, the first line is the sentence to be edited, and the second line is the command string.

Output Format

For each test case, output in one line the resulting sentence. Ignore any space at the end of the sentence.

Sample Input

```
AK47 is powerful
>[2]^[B]>[1]#[48]
Sam sings badly
>[4]<<>[1]#[u]>[3]^[ i]>[6]![2]
Apple is better than Samsung
>[3]<<<<<<^[hTC]![2]
```

Output for the Sample Input

```
AKB48 is powerful
Samsung is bad
hTC is better than Samsung
```

Problem B
Integer Partition

Input File: *pb.in*

Time Limit: *5 seconds*

Problem Description

A new partition problem is defined as follows. Let the symbol $P^i_{y,z}$ be the number of ways to write a positive integer y as a sum of i positive integers having the largest part no larger than z, i.e.,

$$\begin{cases} y = a_1 + a_2 + \cdots + a_i, \text{and} \\ z \geq a_1 \geq a_2 \cdots \geq a_i \geq 1. \end{cases}$$

Notice that two sums differing only in the order of their summands are considered to be the same partition. For example, $P^2_{5,4} = 2$ (can be partitioned in two distinct ways: 4+1, 3+2) and $P^2_{5,3} = 1$ (can be partitioned in one single way: 3+2).

Please write a program to compute the number of $P^i_{y,z}$ with given integers y, i, and z, which y may be as large as up to 500.

Technical Specification

1. $1 \leq y \leq 500$
2. $1 \leq i \leq 30$
3. $1 \leq z \leq 100$

Input File Format

The first line of the input file contains one integer $m(\leq 5)$ indicating the number of test cases to follow. In each of the following m lines, there are three integers y, i, and z.

Output Format

For each test case, output $P^i_{y,z}$.

Sample Input

```
5
20 4 7
100 50 51
487 18 87
492 19 89
500 19 90
```

Output for the Sample Input

```
13
204226
7139824136004762
20430740394679891
25985433353057732
```

Problem C
Matrix
Input File: *pc.in*
Time Limit: *5 seconds*

Given k sets, where $k \geq 2$ and each set has n integers, they are said to be *compatible* if a $k \times n$ matrix B can be constructed from them to meet the following two conditions:

- Each row of B is constructed from a different set and is a permutation of the integers in the corresponding set.
- For all i and j, $1 \leq i \leq k-1$, $1 \leq j \leq n$, $b_{i,j}$ is less than $b_{i+1,j}$, where $b_{i,j}$ is the element at the ith row and jth column of B.

For example, consider two sets $\{6, 3, 5\}$ and $\{1, 4, 2\}$. The two sets are compatible because a 2×3 matrix B can be constructed as follows to meet the above two conditions. The first row of B is from the second set and is [1 4 2], while the second row of B is from the first set and is [3 6 5].

Now consider m ($m \geq 2$) sets each of which has n integers. Also assume that among the m sets, at least two of them are compatible. There are many possible ways to select k ($2 \leq k \leq m$) sets from the m sets. Let k_{max} denote the largest number of sets which are selected from the m sets and are compatible. Your task is to write a program that computes k_{max}.

Technical Specification

- The number of sets, m, is at least 2 and at most 2500. At least two of the m sets are compatible. The number of integers in each of the m sets, n, is at least 1 and at most 20. Each integer in a set is at least 1 and at most 50000.

Input File Format

The first line of the input file contains an integer, denoting the number of test cases to follow. For each test case, the first line contains two positive integers m and n, respectively denoting the number of sets and the number of integers in each set; in the next m lines, each line gives n integers for a set.

Output Format

For each test case, output its k_{max} in a new line.

Sample Input

```
2
2 3
6 3 5
```

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

Output for the Sample Input

```
2
3
```

Problem D
A Matrix Decipher

Input File: *pd.in*
Time Limit: *3 seconds*

Problem Statement

Let $Z_N = \{0,\ 1,\ 2,\ \cdots,\ N-2,\ N-1\}$, where N is a positive integer. An integer linear transformation under ($mod\ N$) can be defined as

$$f(\mathbf{x}) = H\mathbf{x}, \quad where \ \ \mathbf{x} \in Z_N^d, \quad and \ \ H \in {Z_N}^{d\times d}$$

That is, $\mathbf{x}$ is a *d-dimensional* column vector and H is a d by d matrix whose elements are from Z_N.

For $d = 3$, we have $f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = H \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, where $h_{ij} \in Z_N,\ 1 \le i,j \le 3$. The inverse integer transformation exists only if $gcd(det(H), N) = 1$, that is, the determinant of matrix H is relatively prime to N. This problem assumes that $N = 11$.

Let Ω={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, : } be the set of 11 characters represented as numbers 0, 1, $\cdots$, 10, respectively. A matrix encipher takes a message, a character string consisting of 12 characters from Ω={0, 1, 2, $\cdots$, 8, 9, : }, as input and outputs an enciphered message of the same length based on applying an integer linear transformation successively on the d characters in each group from the input message string. For example, for $d = 3$, a message "9870::", decomposed as two groups of 3 characters, "987" and "0::", is enciphered as "262976" based on the transformation matrix $H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}$. The corresponding decipher takes "262976" as input associated with the integer transformation matrix $H^{-1} = \begin{bmatrix} 2 & 10 & 0 \\ 10 & 2 & 10 \\ 0 & 10 & 1 \end{bmatrix}$ which converts "262976" back to "9870::". This problem asks you to design a *matrix decipher* H^{-1} based on a given *matrix encipher* H to decrypt an enciphered message.

Input File Format

The first line of the input file always contains one integer indicating the number of test cases to come. *Each of the test cases* consists of $d+2$ lines with $d = 2$ or $d = 3$ in the first line indicating the dimension of the encipher matrix H followed by d lines of the entries of H row by row, the $(d+2)$-*th* line is the input message of 12 characters.

Output Format

The output format is similar to the input format. The first line of the output file contains one integer indicating the number of test cases. *Each of the test cases* consists of $d+2$ lines with $d = 2$ or $d = 3$ in the first line indicating the dimension of the decipher matrix H^{-1}, the next d lines are the entries of H^{-1} row by row, and the $(d+2)$-*th* line is the decrypted message of 12 characters.

Sample Input

```
   2
   2
   2   1
   3   2
6:19278694:2
   3
   1   1   1
   1   2   2
   1   2   3
26242669:976
```

Sample Output

```
   2
   2
   2  10
   8   2
224488::3377
   3
   2  10   0
  10   2  10
   0  10   1
9876543210::
```

Problem E
Hostage Rescue
Input File: *pe.in*
Time Limit: *15 seconds*

Commander, we are in an emergent hostage situation! Jason Bourne, one of our best special agents, got caught and imprisoned in a building with very high level security control. Now we have two options. Option one, the government pays the ransom money and Bourne might be released. Option two, you, as the leader of the team of special agents, lead the team to get into the building and rescue Bourne.

The information said that the building is equipped with a laser monitoring system. This system consists of a lot of laser devices, which are controlled by two sets of switches. Each switch has two states, on and off. Each laser device is controlled by one switch in the *red* set and one or two switches in the *green* set. The laser device can be turned on by only one combined state of the two or three controlling switches, for example, {on, off, off}. All other states of the switches will turn off the device.

The only way to rescue Bourne safely from the building is to turn off all laser devices in the monitoring system. By some way Bourne sent us the code book of the relationship between the laser devices and switches. The code for a laser device is a tuple. The first element is a letter, starting from A, to indicate the switches in the red set. The upper (lower) case of the letter means the switch must be on (off) to turn on the laser device. The remaining two elements are integers, starting from 1, to indicate the switches in the green set. The sign $+$ ($-$) refers to that the switch must be on (off) to turn on the laser. More precisely, the code book is interpreted like this:

- A code "$A + 2 - 3$" means that the state {on, on, off} of the first switch in the red set, the second, and the third switches in the green set will turn on one laser device.
- A code "$b - 4 + 7$" means that the state {off, off, on} of the second switch in the red set, the fourth, and the seventh switches in the green set will turn on one laser device.

Now we know how each laser device is turned on. (Of course, we know how to turn them off.) You should justify if we have any opportunity of turning off all laser devices so that we can rescue Bourne safely. Otherwise, the government must pay a huge amount of ransom money to get him back.

Technical Specification

- The number of switches in the red set is R, $1 \leq R \leq 10$.
- The number of switches in the green set is G, $1 \leq G \leq 100$.
- The number of laser devices in the monitoring system is L, $1 \leq L \leq 500$.

Input File Format

The first line of the input file contains an integer, denoting the number of test cases to follow. For each test case, the first line contains three integers for R, G, and L, separated by a space. Each of the following L lines contains one character and two integers. Each integer is preceded by a sign symbol.

Output Format

For each test case, output Y if it is possible to turn off all laser devices. Otherwise, output N. Output the result for each test case in a separate line.

Sample Input

```
3
1 1 3
A +1 +1
a +1 +1
A -1 -1
1 1 4
A +1 +1
a +1 +1
A -1 -1
a -1 -1
2 2 9
A +1 +2
a -1 -1
b +1 +2
b +2 +2
a -2 -2
A -1 +2
A -2 -2
a +1 +1
b -2 -2
```

Output for the Sample Input

```
Y
N
N
```

Problem F
Optimal Transformation Cost

Input File: *pf.in*
Time Limit: *3 seconds*

A mathematician, Professor Lee, is now studying a transformation scheme in Coding Theory. There are 2^n n-bit binary strings $S = \{b_n b_{n-1} \ldots b_i \ldots b_2 b_1 |\ b_i \in \{0, 1\}$ for $1 \le i \le n\}$. Two strings, can be transformed each other if and only if one is $\underbrace{b_n b_{n-1} \ldots b_{k+1}}_{n-k} 0 \underbrace{b_{k-1} b_{k-2} \ldots b_1}_{k-1}$ and the other is $\underbrace{b_n b_{n-1} \ldots b_{k+1}}_{n-k} 1 \underbrace{b_{k-1} b_{k-2} \ldots b_1}_{k-1}$, where $i = 0, 1$ and $1 \le k \le n$ (i.e., their binary strings differ in a one-bit position only). We use $x \leftrightarrow y$ to denote the transformation between two strings x and y, and use $cost(x, y)$ to denote the *cost* of the transformation $x \leftrightarrow y$. To make the problem much easier, we assume the cost of each transformation is a constant c.

Professor Lee aims at finding a sequence of transformations $\mathcal{T}(S) = \langle s_1 \leftrightarrow s_2 \leftrightarrow s_3 \leftrightarrow \cdots \leftrightarrow s_{m-1} \leftrightarrow s_m (= s_1)\rangle$ among S such that the following two conditions hold:

1. Every possible transformation is contained at least once by $\mathcal{T}(S)$.
2. The transformation cost of $\mathcal{T}(S)$, defined by $\sum_{i=1}^{m-1} cost(s_i, s_{i+1})$, is as smallest as possible.

The minimum transformation cost of $\mathcal{T}(S)$ is called the *optimal transformation cost*, denoted by $cost(\mathcal{T}(S))$. For example, consider $S = \{000, 001, 010, 011, 100, 101, 110, 111\}$ and assume that $c = 1$. Then, $\mathcal{T}(S) = \langle 000 \leftrightarrow 001 \leftrightarrow 011 \leftrightarrow 010 \leftrightarrow 000 \leftrightarrow 100 \leftrightarrow 101 \leftrightarrow 001 \leftrightarrow 101 \leftrightarrow 111 \leftrightarrow 011 \leftrightarrow 111 \leftrightarrow 110 \leftrightarrow 010 \leftrightarrow 110 \leftrightarrow 100 \leftrightarrow 000\rangle$ and $cost(\mathcal{T}(S)) = 16$. Note that $\mathcal{T}(S)$ may not be unique, but $cost(\mathcal{T}(S))$ is unique.

Given a positive integer n and the cost of each transformation c, your task is to write a computer program to calculate the optimal cost $cost(\mathcal{T}(S))$.

Technical Specification

- $2 \le n \le 20$.
- $1 \le c \le 100$.

Input File Format

The first line of the input file contains an integer, denoting the number of test cases to follow. For each test case, one line contains two integers n and c separated by a space.

Output Format

For each test case, output the optimal cost.

Sample Input

```
2
3 1
2 1
```

Output for the Sample Input

```
16
4
```

Problem G
SHA-4

Input File: *pg.in*
Time Limit: *3 seconds*

This is a new hash algorithm invented for NCPC 2012, called SHA4 (simple hash algorithm version 4). Given a message string M, the SHA4 will hash the string into a 160 bits value, called the message digest d.

d = SHA4(M), where SHA4 is the hash function

We will give you the SHA4 algorithm and several hashed results of message digests: $d_0, d_1, d_2, ...$ You are to write a program to find the original message $M_0, M_1, M_2, ...$ such that

$d_0 = SHA4(M_0), d_1 = SHA4(M_1), d_2 = SHA4(M_2), ...$

Pseudocode for the SHA4 algorithm is listed in the following:

Algorithm 1: SHA4

```
Input: M[0..4]: five characters
Output: digest: five 32 bits hexadecimal values
/* initialized variables:  0x means the values are in hexadecimal
   format */
$h0 \leftarrow 0xdead$; $h1 \leftarrow 0xcafe$; $h2 \leftarrow 0xbeef$; $h3 \leftarrow 0x3399$; $h4 \leftarrow 0x7788$;
$a \leftarrow h0$; $b \leftarrow h1$; $c \leftarrow h2$; $d \leftarrow h3$; $e \leftarrow h4$;
for $i \leftarrow 0$ to $4$ do
    $word[i] = M[i] -$ ' ';
    /* The ASCII value of space is subtracted from the input character
       */
    $f \leftarrow b + c$; $k \leftarrow 0x5a82$;
    $temp \leftarrow 4 * a + f + e + k + word[i]$;
    $e \leftarrow d$; $d \leftarrow c$; $c \leftarrow 8 * b$; $b \leftarrow a$; $a \leftarrow temp$;
$h0 \leftarrow h0 + a$; $h1 \leftarrow h1 + b$; $h2 \leftarrow h2 + c$; $h3 \leftarrow h3 + d$; $h4 \leftarrow h4 + e$; /* produce the
   final hash value:  */
digest $\leftarrow h0$ append $h1$ append $h2$ append $h3$ append $h4$;
/* C like code:  */
/* printf("%08x %08x %08x %08x %08x", $h0, h1, h2, h3, h4$) */
```

Note:

1. All variables are unsigned 32 bits integers.
2. The input string **M** is with length 5.

3. Each input character **M[i]** is converted into an integer value, by subtracting ASCII value of space from the ASCII value of **M[i]**, that is, the difference between space and the character ASCII value. For example, the space value is 0.
4. After the value conversion, the five bytes values are stored in word[i], $0 \le i < 5$.

Technical Specification

- The number of test cases is less than 1024.
- All output strings are with length 5.

Input File Format

The first line of the input file contains an integer, denoting the number of test cases to follow. For each test case, the message hash digest is given with values of five 32 bits integer in hexadecimal format, each integer separated with at least one space. For example, a sample digest is 0b8414f6 027eeb13 0453edaf 00f93379 002f2d88. All digest values are valid SHA4 outputs.

Output Format

For each test case, output the original message with one line of string, containing only five alphanumeric characters (A-Za-z0-9). For example, you are to output '5dogs' if the input is SHA4('5dogs')='0b8414f6 027eeb13 0453edaf 00f93379 002f2d88'.

Sample Input

4
0b8414f6 027eeb13 0453edaf 00f93379 002f2d88
0b8419b0 027eec17 0453ef3f 00f933f1 002f2da8
0b8459a5 027efa02 045406ff 00f93981 002f2f10
0b845578 027ef91a 04540577 00f93921 002f2ef8

Output for the Sample Input

```
5dogs
9cats
fade4
cafe8
```

Problem H
Gap Verification
Input File: *ph.in*
Time Limit: *5 seconds*

There is a software industry called EDA (Electronic Design Automation) which produces software tools for IC hardware engineers to help manufacturing ICs. Manufacturing a chip must go through several phases. Each phase may need help from several software tools. Fig. 1 shows the output from one of these software tools. In this phase, circuits and transistors (i.e., PNP transistors or NPN transistors) are rendered as rectangles on different layers of materials.

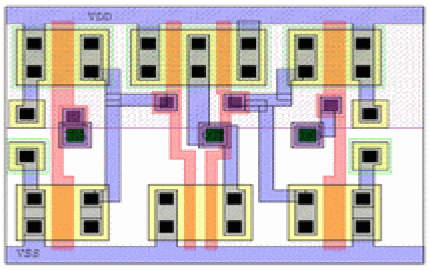

Figure 1: The IC layout produced by an EDA software tool.

However, due to the limitation of layout algorithms, these layouts need additional verification and manual adjustment. One rule to check is the spacing between two rectangles (see Fig. 2). If the spacing is less than a minimum distance, two rectangles may become connected during wafer's micro-fabrication process. Given a set of rectangles and a spacing rule, please write a program to find out the places that violates the spacing rule horizontally.

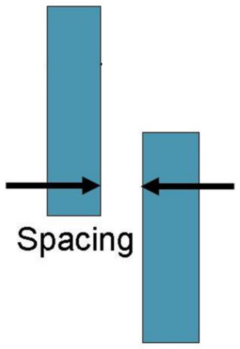

Figure 2: The spacing rule.

Fig. 3 are some boundary cases for your references. Please assume the IC layout algorithms are still good enough so that rectangles never overlap themselves. In Fig. 3 (a), two rectangles are aligned at a same Y-coordinate. In this case, the spacing rule needs to be checked. In Fig.3(b), two rectangles do not overlap at Y-axis, so, you don't need to check spacing rule at X-axis for the two rectangles. Although rectangles do not overlap, they can

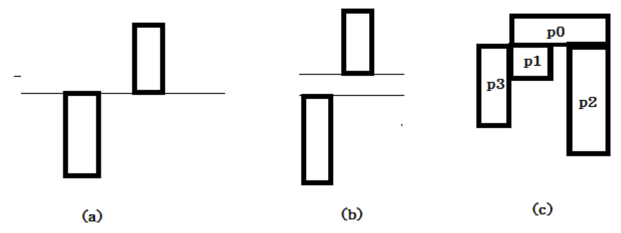

Figure 3: Boundary cases.

be connected as in Fig. 3(c). In this case, your program must understand *p0-p1-p2-p3* belong to a same region. So, you don't need to report the violation of spacing rule between *p3-p1, p3-p0, p1-p2*, and *p3-p2*.

1. Please ignore the cases where two rectangles only touch at corner alone (such as *p0* and *p3* alone).

2. You only need to check the spacing rule horizontally (i.e., X-axis)

Input Format

The test data begins with an integer $N (N < 10)$, which is the number of test cases. In each test case, it begins with two integers P and S. P is the number of rectangles ($P \leq 50001$) and $S (0 < S < 1000)$ is the minimum spacing distance that your program needs to check. If the distance of two rectangles is less or equal than S, the violation must be reported. Following is P lines of rectangle information. Each rectangle is described by (*index x y width length*). index (an integer in the range from 0 to $P - 1$) is identification of a rectangle. (x, y) is the top-left corner coordinates (2D plane coordinates which can be held by long integers). *width* is the size of a rectangle on X-axis and *length* is the size of a rectangle on Y-axis.

Output Format

For each test case, please output the places where two rectangles violate the spacing rule in X-axis. The violation place is output as "($p1$ $p2$)" where $p1 < p2$ are the indexes of the rectangles. If there are multiple violations to report, please sort the violations by $p1$ first in ascending order. Then please sort the violations with same $p1$ by $p2$ in ascending order. There is no space between two violation places.

Sample Input

```
2
5 15
0 0 0 10 20
1 10 30 30 10
2 10 20 10 10
```

```
3 30 20 10 20
4 20 -10 10 10
5 15
0 0 0 10 20
1 10 50 30 10
2 10 20 10 10
3 30 20 10 20
4 20 -10 10 10
```

Sample Output

```
(0 4)
(0 4)(2 3)
```

Problem I
Christmas Gifts
Input File: *pi.in*
Time Limit: *3 seconds*

Children always expect gifts at Christmas Eve. Every child in the community No-Common-Present-Compromise (NCPC for short) can have two wishes for toys, and Santa Claus always promises at least one of the two wishes. Children in NCPC are all unselfish such that a toy can be shared by many children. This year, all the wishes have been collected and you are asked to help Santa Claus to prepare the gifts for all the children. Due to the limited capacity of the cart, you need to minimize the number of gifts. But remember, for every child, at least one of the two wishes should be satisfied.

For the example shown in the first test case of the **Sample Input**, there are four children. The first child wishes to have toy 0 or 1, and the toys wished by the other three children are respectively 100 or 1, 100 or 0, and 100 or 200. For this case, you can prepare only two toys (0 and 100) to satisfy all the four children.

Technical Specification

1. Every wish is a toy and labeled by an nonnegative integer at most 10000.
2. The number n of different toys is at most 250.
3. The number m of children is a positive integer and at most 5000.
4. For each test case, $n \leq 100$ or $m \leq 400$.

Input File Format

The input file contains several test cases. For each test case, the first line contains the integer m which is the number of children in this case. In the following m lines, each line contains two integers separated by a space, in which the two integers are the toy labels the child wishes for. A case with $m = 0$ indicates the end of the input and you don't need to process it.

Output Format

For each test case, output in one line the minimum number of gifts that Santa Claus should prepare.

Sample Input

```
4
0 1
```

```
100 1
100 0
100 200
12
10 20
20 30
30 40
40 50
50 60
60 70
70 80
80 10
10 0
30 0
50 0
70 0
0
```

Output for the Sample Input

```
2
4
```

Problem J
The Resequencing LCS Problem

Input File: *pj.in*

Time Limit: *15 seconds*

Let $X = x_1x_2x_3 \cdots x_m$ and $Y{=}y_1y_2 \cdots y_n$ be two strings over a finite alphabet set Σ. A *subsequence* of a string is obtained by deleting zero or some (not necessarily consecutive) characters in this string. A *common subsequence* of X and Y is a subsequence occurring in both X and Y. A *longest common subsequence* (LCS for short) of X and Y is a common subsequence of X and Y with the maximum length.

Given a set $S = \{S_1, S_2, \ldots, S_\ell\}$ of ℓ strings, a text T, and a natural number k, find a string M, which is a concatenation of k strings (not necessarily distinct, i.e., a string in S may occur more than once in M) from S, whose length of LCS with T is largest. Such a string is called a k-inlay. The resequencing LCS problem is to find the length of the LCS between T and a k-inlay for each query with parameter k after T and S are given.

For example, let $S = \{agc, act, aatg, ttcg\}$ and $T = agactagtc$ in which $S_1 = agc, S_2 = act, S_3 = aatg$, and $S_4 = ttcg$. If $k = 2$, then both $S_1S_1 = agcagc$ and $S_1S_4 = agcttcg$ are 2-inlays, and the length of their LCSs with T is 6. For $k = 4$, $S_1S_2S_1S_4 = agcactagcttcg$ is a 4-inlay, and the length of its LCS with T is 9.

Technical Specification

- The alphabet set Σ contains all 26 English alphabets, i.e., $\{a, b, \ldots, z\}$.
- The number of characters in T is no more than 500.
- The number of strings in S is no more than 50 and the length each string in S is no more than 50.
- The number of queries is no more than 10 and, in each query, $k \leqslant 30$.

Input File Format

The first line of the input file contains an integer, denoting the number ℓ of strings in S. The second line of the input file contains an integer, denoting the number n of queries. The third line of the input file contains the string T. In the following $\ell + n$ lines, the ith line contains a string S_i for $i = 1, 2, \ldots, \ell$. The $(\ell + i)$th line contains the value of k for the ith query for $i = 1, 2, \ldots, n$.

Output Format

For each query, output the length of its LCS with T on one line.

Sample Input

```
4
```

```
2
agactagtc
agc
act
aatg
ttcg
2
4
```

Output for the Sample Input

```
6
9
```