2012 National Collegiate Programming Contest

- Problems: There are 6 problems (11 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 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.
^[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
```