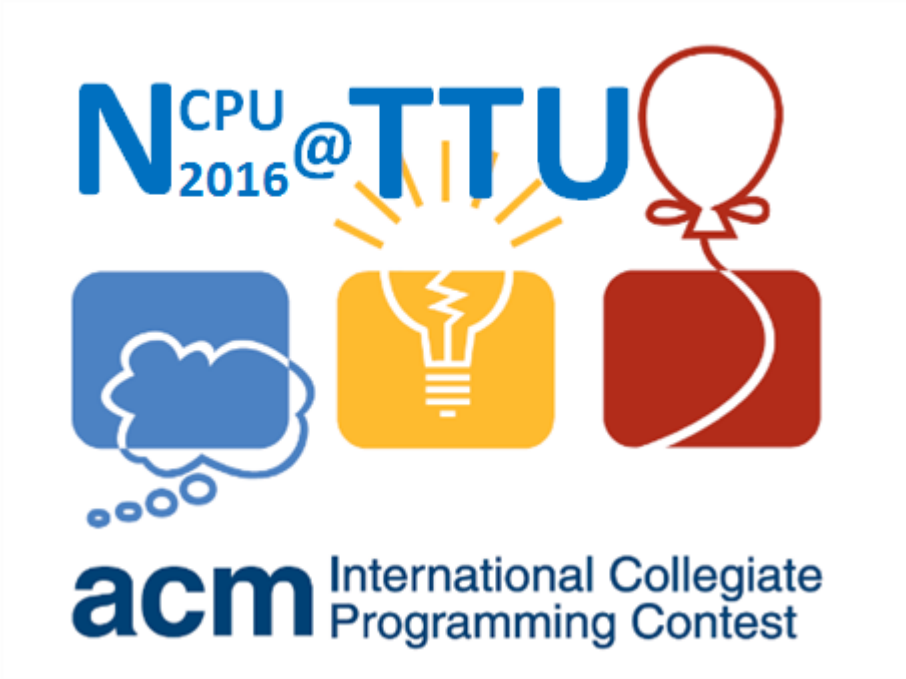
第六屆 ACM ICPC

全國私立大專校院程式競賽

National Contest for Private Universities,

Taiwan 2016



競賽題目

Problem 1: Maximum Subsequence Sum Problem

(Time Limit: 3 seconds)

Given a sequence containing both negative and positive integers. Find the contiguous subsequence with maximum sum. For example, for input {-2, 11, -4, 13 -5, -2}, the answer is 20 (the 2nd through the 4th). If all numbers are negative, the result is maximum of them, i.e., smallest in terms of absolute value.

Input

The input begins with a single positive integer T ($1 \leq T \leq 20$) on a line by itself indicating the number of test cases following, each of them as described below.

The first line of each test case contains a single integer N ($1 \leq N \leq 10000$) that indicates how many numbers are in the sequence. The second line contains N space-separated integers. All these numbers are between -10000 and 10000.

Output

For each test case, output one line of three space-separated integers: the maximum sum, the starting position, the ending position of the contiguous subsequence, respectively. If there is more than one such subsequence, output the one that has minimum length. If there is more than one subsequence of minimum length, output the one that occurs first.

Sample Input

```
2
6
-2 11 -4 13 -5 -2
4
-2 -4 -1 -3
```

Sample Output

```
20 2 4
-1 3 3
```

Problem2: Unlocking Steps

(Time Limit: **3** seconds)

Problem Description

A combination lock is a locking device where a sequence of numbers is used to open the lock. The sequence is entered by rotating discs with inscribed numbers. Each disc is inscribed with following digits, $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$, $\mathbf{3}$, $\mathbf{4}$, $\mathbf{5}$, $\mathbf{6}$, $\mathbf{7}$, $\mathbf{8}$, and $\mathbf{9}$, and the digit next to $\mathbf{9}$ is $\mathbf{0}$. Each disc can be rotated clockwise or counter clockwise. An unlock step is a neighboring digit rotation. For example, rotating one disc from $\mathbf{1}$ to $\mathbf{2}$ costs $\mathbf{1}$ step. Rotating one disc from $\mathbf{0}$ to $\mathbf{9}$ also costs $\mathbf{1}$ step. There may be several digits on a lock. Given an initial number of a combination lock $\mathbf{11677}$, the minimum number of steps to unlock the lock to the key number $\mathbf{04405}$ is $(\mathbf{1}-\mathbf{0})+(\mathbf{4}-\mathbf{1})+(\mathbf{6}-\mathbf{4})+(\mathbf{10}-\mathbf{7})+(\mathbf{7}-\mathbf{5})=\mathbf{1}+\mathbf{3}+\mathbf{2}+\mathbf{3}+\mathbf{2}=\mathbf{11}$.

Input Format

The first line is a number $\boldsymbol{n}$ indicating the number of test cases. Each test case has **2** numbers in one line to represent the initial number and the key number of the lock.

Output Format

For each test case, output the minimum number of steps to unlock the lock.

Technical Specification

- Number $\boldsymbol{n}$ is an integer. $\mathbf{1} \leq \boldsymbol{n} \leq \mathbf{10}$.
- The length of initial number or key number is $\mathbf{9}$ at most.

Example

Sample Input:	**Sample Output:**
3 1234 1284 000000 123456 319 520	5 19 4

Problem3: Division of Polynomials

(Time Limit: 3 seconds)

A polynomial may be represented by two sequences, one contains the degrees in descending order of every term in the polynomial (zero-coefficient terms are not included) and the other contains the corresponding coefficients. For example, a polynomial $6x^3 + 5x^2 - 7$ is represented as {3, 2, 0} and {6, 5, -7}. Given two polynomials, print the resultant quotient and remainder polynomials after division of the two polynomials. The divisor is a monic polynomial in which the leading coefficient (the nonzero coefficient of highest degree) is equal to 1.

Input

The input begins with a single integer T ($1 \leq T \leq 100$) on a line by itself indicating the number of test cases following, each of them as described below. For each test case, the first line contains each polynomial term's degree ($0 \leq d \leq 100$) in descending order, the second line contains each term's coefficient ($-100 \leq c \leq 100$), and the last two lines form the divisor polynomial. All these numbers are integer.

Output

For each test case, print the paired sequences in separate lines (degree's first, then coefficient's) that represent resultant quotient polynomial, then similarly print the paired sequences that represent resultant remainder polynomial. Notice that either the quotient or the remainder polynomial may be empty (represented as {0} and {0}).

Sample Input

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

Sample Output

```
0
0
1 0
1 1
1 0
6 7
1
20
```

Problem 4: Multiplicative Cipher

(Time Limit: 3 seconds)

Problem Description

The multiplicative cipher is a type of substitution cipher where each letter of the plaintext is substituted with another letter to form the ciphertext. In the multiplicative cipher, encrypting and decrypting letters involved converting them to numbers, multiplying the key, and then converting the new number back to a letter.

Your task is to write a program to decryption message using the multiplicative cipher. That is, given the key of the cipher and ciphertext, your program must output the plaintext. The plaintext/ciphertext contained only plus sign (+), uppercase letters (A to Z) and digits (0 to 9). The plaintext/ciphertext has been encrypted/decrypted via the following method.

Encryption

- Convert the plaintext letters to numerical values as follows:
  '+' => 0, 'A' => 1, 'B' => 2, … , 'Z' => 26,
  '0' => 27, '1' => 28, '2'=> 29, … , '9' => 36.

- The encryption function for a single letter is

$$C = k \times P \pmod{37}.$$

  - P means the number corresponding to plaintext letter.
  - C means the number corresponding to ciphertext letter.
  - k is the key of the cipher.
  - The value k must be chosen such that k and 37 are coprime.

- Convert the numerical values to ciphertext letters as follows:
  0 => '+', 1 => 'A', 2 => 'B', … , 26 => 'Z',
  27 => '0', 28 => '1', 29 => '2', … , 36 => '9'.

Decryption

- Convert the ciphertext letters to numerical values as follows:
  '+' => 0, 'A' => 1, 'B' => 2, … , 'Z' => 26,
  '0' => 27, '1' => 28, '2'=> 29, … , '9' => 36.

- The decryption function for a single letter is

$$P = k^{-1} \times C \pmod{37}.$$

  - k^{-1} is the modular multiplicative inverse of a modulo 37.
  - That is, $k \times k^{-1} = 1 \pmod{37}$.

- Convert the numerical values to plaintext letters as follows:
  0 => '+', 1 => 'A', 2 => 'B', … , 26 => 'Z',
  27 => '0', 28 => '1', 29 => '2', … , 36 => '9'.

Example with $(k, k^{-1}) = (25, 3)$:

Plaintext letters:	A	B	C	D	1	2	3	4
Plaintext numbers:	1	2	3	4	28	29	30	31
Ciphertext numbers:	25	13	1	26	34	22	10	35
Ciphertext letters:	Y	M	A	Z	7	V	J	8

Input Format

The first line of input indicates the number of test cases, T ($1 \leqq T \leqq 100$). For each case, the first line of input is the given cipher key, k ($1 \leqq k \leqq 36$). The next line of input is the ciphertext which is composed by one line and cannot exceed 70 characters of length.

Output Format

For each test case, the output should be a single line containing the original plaintext.

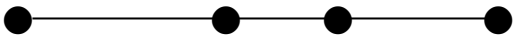
Example

Sample Input:	**Sample Output:**
2 30 RFX+3VB+JFA+66U 19 DSHH4	HOW+ARE+YOU+007 HAPPY

Problem 5: Set Operations On Segments

(time limit: 2 seconds)

You are given two sets of horizontal line segments on the X-axis. You are to compute the intersection and union of the two sets. In mathematics, a closed interval is defined as all the points between two endpoints and includes the endpoints. The intersection of two segments is defined as the closed interval where the two segments overlap. It follows from this definition that the intersection cannot be a single point; it must either be another segment or nothing. The union of two segments is defined as the smallest closed interval that contains both segments if and only if the two segments overlap or meet at an endpoint. Otherwise, the union results in the two original segments. Graphically,



$\overline{AB} \cap \overline{BC} = \emptyset$; $\overline{AC} \cap \overline{BC} = \overline{BC}$

$\overline{AB} \cup \overline{BC} = \overline{AC}$; $\overline{AC} \cup \overline{BD} = \overline{AD}$; $\overline{AB} \cup \overline{CD} = \overline{AB}$ and $\overline{CD}$

Input

The first line of input is an integer K, $K \leqq 10$, that specifies the number of test cases for the problem. The rest of input is divided into two parts. The first part begins with a series of assignment statements, one statement per line terminated by the end-of-line character, of the following format:

English alphabet=floating-point number

For example, A=3.457. The floating-point number is the coordinate of a point denoted by the alphabet. You can be certain there is no space before or after the assignment statement, and note that there is no space before or after the equal sign. This part of input is terminated by a line that contains two equal signs. The second part of input consists of K test cases, each case on two lines, and the test cases are separated again by a line with two equal signs. Each of the two lines specifies a set of segments, with each segment defined by giving its endpoints that are two alphabets from the previous part of input. There is no space within each group of two alphabets, and a blank separates consecutive groups. Each set of segments is terminated by the end-of-line character. You can be sure that segments from the same set do not overlap. However, segments which are joined at endpoints must be treated as a single segment.

Output

Output is on two lines for each test case. The first line displays the intersection and the second line displays the union of the two sets. Each line consists of segments that result from a set operation. These segments must be listed from left to right, and each segment is specified by a pair of alphabets, with the alphabet of the left endpoint shown first followed by the alphabet of the right endpoint. As in the input, there is no space within a pair of alphabets, and a space separates two consecutive pairs. If the outcome of the intersection of the given sets is nothing, output "empty" on a single line.

Sample Input

```
2
A=2.3
a=4.57
B=3.5
b=1.4
C=1.28
D=7.21
f=0.97
E=0.2
g=10.88
d=6.32
==
CB DB gD fE
Ab Ba dg
==
aA dD
gD da
```

Sample Output

```
bA Ba dg
Ef Cg
empty
Ag
```

Problem 6: Maximum Zero-Crossing Window

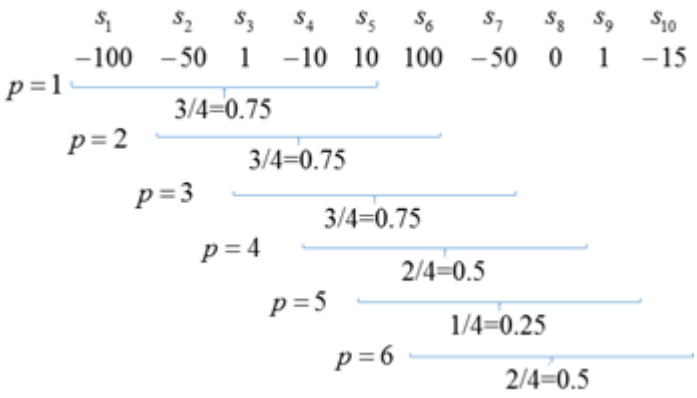
(Time Limit: 3 seconds)

The zero-crossing rate (ZCR) is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval, being a key feature to classify percussive sounds.

In this problem, you will be given a sequence of signal strengths of length n, say, $s_1, s_2, \ldots, s_n$ and a window length w. The zero-crossing rate zcr_p for such a window at location p is defined by

$$zcr_p = \frac{1}{w-1}\sum_{i=1}^{w-1} I\left(s_{p+i-1}s_{p+i} < 0\right)$$

where $I(\cdot)$ is an indicator function whose value is 1 if and only if its argument is true. Your task is to find the location p whose corresponding window has the maximum zero-crossing rate. If there are more than one such locations, output the one with minimum index. For example, consider the following signal of length 10, and window length 5. The maximum zero-crossing rate window is located at position 1.



Input

The first line of input indicates the number of test cases T. These is a blank line before the test data of each case. The test data of each case spans several lines. The first line contains two integers n ($2 \le n \le 100000$) and w ($2 \le w < n$), and the next serval lines

contains n white character separated integers representing the signal strengths whose values are within -2147483648 and 2147483647.

Output

For each test case, output one line of a single integer, which represents the position of the maximum zero-crossing window. If there are more than one such window, output the one with minimum index.

Sample Input

```
2

10 5
-100 -50 1 -10 10
100 -50 0 1 -15

12 6
0 100 -1 -5 -4 30 -9 -9
10 0 31 -8
```

Sample Output

```
1
2
```

Problem 7: Breaking Bad Ransomware

(Time Limit: 3 seconds)

Problem Description

Cryptolocker is a kind of ransomware to lock victims' valuable files from access by a strong cryptographic algorithm for data encryption. The scheme used is usually a public system and even the FBI has told the innocent users that the quick way to recover the files is to pay the ransom. However, the government authorities have been commanded to form a task force named: National Cryptolocker Protection Unit (NCPU in short) to release the innocence. They have found a flawed implementation of cryptolocker You are asked to decrypt the files.

This implementation is an offline version of crytolocker. That is, the key is stored in the victim's disk using a sorting scheme based on the prime factorization. The scheme called prime factorization sorting (PFS in short) is described as follows.

PFS: Numbers in the range of 1 to n must be sorted in terms of the number of factors in their prime factorization. In case of a tie, the smaller number will come first. For example, 20 = 2*2*5, so it has 3 numbers in its prime factorization. Similarly 35 = 5*7 has 2 numbers in its prime factorization. Therefore, 35 will come before 20 according to this criterion.

Given a key with *m* numbers and the plain text to be encrypted (a string), divide the plain text into substrings of length ***m*** and the characters of each substring are reordered to create the cipher text. The reordering is based on how the sorted key corresponds to the original key.

For example, let's assume that the key is "20 40 30 10 50 50". The PFS sorted version of the key is "10 20 30 50 50 40". The six numbers in the sorted version have indices 4-1-3-5-6-2 in the original key. Each substring in the plain text is reordered accordingly. For example, a substring "number" is reordered and encrypted to "bnmeru". Given the encrypted string: "bnmeru" and the key: "20 40 30 10 50 50", you are to output the decrypted string: "number".

When there are duplicate numbers in the key, their relative orders stay the same after sorting. For the key above, this means that the indices are 4-1-3-5-6-2 and not 4-1-3-6-5-2.

Let the last substring in the plain text has length k. If k<m, reorder the last substring using just the first k characters in the original key as its key.

Input Format

The first line is the number of test cases ***T***. The next two lines are the key with ***m*** numbers $n_1\ n_2\ \ldots\ n_m$ separated with at least one space and the encrypted text. The maximum length of each encrypted string is ***L***. The character of the string is restricted in "0-9A-Za-z". This is repeated for all the test cases.

Output Format

Each line should be the decrypted text of the encrypted string in a test case.

Technical Specification

- Number ***T*** is an integer. $\mathbf{1} \leq \boldsymbol{T} \leq 20$.
- Numbers ***L*** is an integer. $\mathbf{1} \leq \boldsymbol{L} \leq \mathbf{100}$.
- Numbers ***m*** is an integer. $\mathbf{1} \leq \boldsymbol{m} \leq \mathbf{20}$.
- Number $n_1, n_2, \ldots, n_m$ are integers. $\mathbf{1} \leq n_0, n_1, n_2, \ldots, n_m \leq 2^{32}$.

Example

Sample Input:	**Sample Output:**
3 1 2 3 4 5 testtest 20 40 30 10 50 50 bnmeru 1826606452 1818930412 60118698 edrlubpiekn	testtest number redbluepink

Problem 8: Multiple shortest routes

(Time Limit: 3 seconds)

Problem Description

There are n buildings, numbered from 0 to $n-1$, and m unidirectional roads. Without passing through any intermediate building, each road starts at one building and ends at another. Alice, who wants to go from building 0 to a destination building, knows the exact amount of time needed to go through each road. Please help her determine whether she has two or more quickest routes to choose from. Unless she has no routes at all to her destination, please also find the time needed to go along one of her quickest route(s). You may assume that Alice's destination building is not building 0.

Input Format

The first line is the number of test cases. Each test case is specified as follows: The first three lines are the number n of buildings, Alice's destination (which is a building) and the number m of edges. Each of the next m lines specifies a road by its starting building, its ending building and the amount of time (which is a positive integer) to go through it, in that order. Consecutive numbers in a line are separated by a space.

Output Format

For each test case, please output two lines. The first line is "yes" if there are two or more shortest routes to go from building 0 to Alice's destination, and it is "no" otherwise. If there is at least one route from building 0 to Alice's destination, then the second line is the minimum amount of time needed to go from building 0 to Alice's destination. Otherwise, it is "x".

Technical Specification

- The number of test cases is at most 15.
- $n \leq 100$ is an integer.
- The amount of time needed to go through any road is a positive integer less than or equal to 100.

Example

Sample Input:	Sample Output:
3 4 2 5 0 1 7 1 2 1 3 1 1 0 3 5 3 2 3 3 2 2 0 1 7 2 1 1 3 2 3 0 1 1 0 2 2 1 2 1	no 7 no x yes 2