XXVII Maratón Nacional de Programación
Colombia - ACIS / REDIS 2013
ACM ICPC

Problems

(This set contains 9 problems; problem pages numbered from 1 to 12)

General Information
Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your source file (your solution!) must be called `<codename>.c`, `<codename>.cpp` or `<codename>.java`, as indicated below each problem's title.

Input

1. The input must be read from standard input.
2. The input contains several test cases. Each test case is described using a number of lines that depends on the problem.
3. When a line of data contains several values, they are separated by single spaces. No other spaces appear in the input. There are no empty lines.
4. Every line, including the last one, has the usual end-of-line mark.
5. The end of input is indicated by the end of the input stream. There is no extra data after the test cases in the input.

Output

1. The output must be written to standard output.
2. The result of each test case must appear in the output using a number of lines that depends on the problem.
3. When a line of results contains several values, they must be separated by single spaces. No other spaces should appear in the output. There should be no empty lines.
4. Every line, including the last one, must have the usual end-of-line mark.
5. After the output of all test cases, no extra data must be written to the output.
6. To output real numbers, round them to the closest rational with the required number of digits after the decimal point. Ties are solved rounding to the nearest lower value.

Problem A
30 Minutes or Less

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

Gridland is a grid-shaped city whose streets are numbered consecutively starting at 1, from left to right and from south to north. The distance between two city locations (x_1, y_1) and (x_2, y_2) is the usual Manhattan distance, i.e., $|x_2-x_1|+|y_2-y_1|$. Pizza Planet, the largest fast food chain in Gridland and the franchise with most restaurants across the city, plans to implement the 30 *minutes or less* guarantee in its delivery service: either customers receive their pizzas within 30 minutes after placing the order or otherwise the pizza order is on the house.

Your task is to reduce the pizza delivery times, i.e., to minimize the total distance *delivery guys* from Pizza Planet restaurants must travel to make all deliveries. Each pizza order must be delivered and each restaurant can dispatch at most one pizza order.

Input

There are several test cases. Each case begins with a blank-separated pair of integer numbers R and N: the number of restaurants ($1 \leq R \leq 100$) and the number of pizza orders ($1 \leq N \leq R$). Then $R+N$ lines of blank-separated pairs of integers x and y follow ($-10^3 \leq x \leq 10^3$, $-10^3 \leq y \leq 10^3$): each of the first R lines indicates the location of a Pizza Planet restaurant and each of the last N lines indicates the location where a pizza order must be delivered.

The input must be read from standard input.

Output

For each case, print the minimum distance the delivery guys must travel to make all deliveries.

The output must be written to standard output.

Sample input	**Output for the sample input**

```
2 2
1 5
2 1
4 2
4 3
3 2
2 1
4 3
7 4
4 5
5 -1
```
	```
	8
	7
	```

Problem B
Gödel's Dream

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

Once Gödel dreamt about codes. And he foresaw a very interesting puzzle involving finite sequences of bits called h-sequences. An *h-sequence* is a string defined recursively as follows:

$$\begin{aligned}\langle hseq\rangle &::= 0\\ \langle hseq\rangle &::= 1\langle hseq\rangle\langle hseq\rangle\end{aligned}$$

For instance, 0, 11000, and 101100100 are h-sequences but 1, 11001, and 111100100 are not.

Gödel's dream was of a tall order of h-sequence fun because the strings had incomplete information. More precisely, strings did not only contain bits '0' and '1', but they also contained the question mark symbol '?'. The occurrence of a question mark symbol in a string indicates that any bit can go in such a position of the string. For example, the sequence 101?0 can actually have the h-sequence 10100 and the string 10110 (which is not an h-sequence) as instances.

Given a string made of bits, possibly containing question mark symbols, your task is to write a program that computes the maximum number of h-sequences that concatenated together result in an instance of such a string.

Input

The input consists of several test cases, each one defined by a line containing a string s made of characters '0', '1', and '?', whose length is between 1 and 10^4 inclusive.

The input must be read from standard input.

Output

For each test case, output a line with the maximum number of h-sequences that concatenated together result in an instance of s.

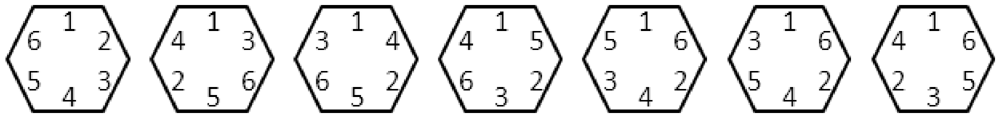
The output must be written to standard output.

Sample input	**Output for the sample input**
`0`	`1`
`10100`	`1`
`??1?`	`0`
`?1??`	`2`
`1?010100`	`2`
`???1????`	`6`

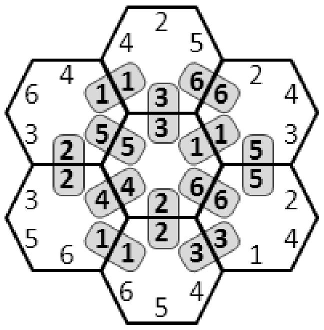
Problem C
Hexagonal Puzzle

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

A *hexagonal piece* is a hexagon whose sides are labeled with distinct integers values between 1 and 6. A *Hexagonal Puzzle* is a set of seven hexagonal pieces, e.g.,



A *Hexagonal Puzzle* is *solvable* if and only if its pieces can be translated and rotated, without reflecting or flipping any of them, to form a honeycomb pattern where neighboring sides of any two pieces are labeled with the same integer values. The following figure depicts a honeycomb pattern witnessing the fact that the above hexagonal puzzle is solvable:



Your task is to determine if a given Hexagonal Puzzle is solvable or not.

Input

The input contains several test cases, each one of them corresponding to the description of a Hexagonal Puzzle. A case comprises seven lines, each one containing a blank-separated permutation of the numbers 1, 2, ..., 6 indicating the clockwise labeling of the sides of a hexagonal piece of the puzzle.

The input must be read from standard input.

Output

For each case print one line with the word “`YES`” if the given Hexagonal Puzzle is solvable, or the word “`NO`” otherwise. Answers should be left aligned.

The output must be written to standard output.

Sample input	**Output for the sample input**

`1 2 3 4 5 6`
`1 3 6 5 2 4`
`1 4 2 5 6 3`
`1 5 2 3 6 4`
`1 6 2 4 3 5`
`1 6 2 4 5 3`
`1 6 5 3 2 4`
`1 2 3 4 5 6`
`1 2 3 4 5 6`
`1 2 3 4 5 6`
`1 2 3 4 5 6`
`1 2 3 4 5 6`
`1 2 3 4 5 6`
`1 2 3 4 5 6`
	`YES`
	`NO`

Problem D
Largest Sum Game

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

After a long beer party and when ready for the check, John and his friends play the *largest sum game* and whoever wins the game will not have to pay a dime of the check. The largest sum game consists in finding the largest sum of consecutive values in a sequence of numbers; the winner is the one quickest to answer.

For example, in the sequence 23, -1, -24, 2, 23 the largest sum of consecutive values is 25 and whoever finds this value first, is the winner of the game.

Although simple (and geeky), the game is challenging because beer and arithmetic do not mix well together. However, since the group of friends are amateur programmers, each have implemented an algorithmic solution for finding the largest sum and have agreed to select the winner of the game in the form of a programming challenge: they connect their laptops to a central server that produces a random sequence of numeric values, run the solutions on this data, and the program quickest to answer wins the game.

John is tired of paying night after night without ever winning the game and he is determined to stop this situation tonight. John has hired you to write a highly efficient computer program that could beat the others in the largest sum game.

Input

The input consists of several test cases, each one defined by a line containing a sequence of N blank-separated integers X_1, X_2, $\ldots$, X_N ($1 \le N \le 10^5$, $-10^3 \le X_i \le 10^3$ for each $1 \le i \le N$).

The input must be read from standard input.

Output

For each test case, output a line with the largest sum of consecutive values in X_1, X_2, $\ldots$, X_N.

The output must be written to standard output.

Sample input	Output for the sample input
`1 2 3 4 5 6 7 8 9`	`45`
`-1 -1 -1`	`0`
`23 -1 -24 2 23`	`25`
`1 -14 -4 14 -11 -7 6`	`14`

Problem E
Reodrnreig Lteetrs In Wrods

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

The following is an excerpt from an unknown but trusted source:

> "Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe."

In this problem, you will be given a dictionary of words and some text. The letters inside each word of the text have been scrambled, but it is known that the first and last letter are in their original place. Your task is to clean up the text by reordering each word in the original text with its correct version from the dictionary.

Input

The first line of the input contains an integer N indicating the number of test cases ($1 \leq N$). Then, N test cases follow. Each test case is described in exactly two lines: the first line contains the list of words in the dictionary, sorted in ascending lexicographical order, and the second line contains the list of words to be cleaned up. Words in both lists are separated by single blanks and are made of just English lowercase letters. You can assume that each list is non-empty and has at most 200 characters, and that the dictionary does not contain duplicates.

The input must be read from standard input.

Output

For each test case, output a single line with the clean version of the text: each word w in the input text should be replaced by a word d in the dictionary such that d can be converted to w by reordering letters except for the first and last ones. If there is more than one word d in the dictionary that could be used for w, replace it with the one that comes first in lexicographical order; if there are no words in the dictionary for w, then take d=w.

The output must be written to standard output.

Sample input	**Output for the sample input**

`2`
`make me programming rich will`
`pagmrnmorig will mkae me rcih`
`dreaming drinaemg yeah`
`yaeh right keep drinaemg`
	`programming will make me rich`
	`yeah right keep dreaming`

Problem F
Shuffling Cards

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

Automated Casino Machines (ACM), a company that manufactures roulettes, slot machines, 6-faced die and the like, has developed the next-generation machine for shuffling cards. This machine offers the feature of shuffling N cards at a time.

The machine works as follows. Initially, each of the N cards is loaded into a slot. There are N slots in total and they are numbered between 1 and N, inclusive. When the machine is activated, the cards are mechanically moved around and as a result each card ends up in a (possibly) different slot. The act of activating the machine once is called a *shuffling round*.

Internally, the machine has a fixed *shuffling function* f that describes the machine's behavior: if a card is currently in slot i, then after one shuffling round it will be placed in slot $f(i)$, for $1 \leq i \leq N$. For example, consider a machine with the shuffling function defined by the following mapping (with $N=5$):

$$1 \mapsto 4 \qquad 2 \mapsto 2 \qquad 3 \mapsto 1 \qquad 4 \mapsto 5 \qquad 5 \mapsto 3$$

This means that after one shuffling round this machine simultaneously moves the cards in slots $1, 3, 4, 5$ to slots $4, 1, 5, 3$, respectively; card in slot 2 is not moved.

Of course, this is not very random because after exactly one round the machine always produces the same result. However, different results can be obtained by activating the machine over and over again without unloading the cards after each intermediate round.

For example, after two shuffling rounds, the aforementioned shuffling machine would simultaneously move the cards in slots $1, 3, 4, 5$ to slots $5, 4, 3, 1$, respectively; card in slot 2 would still not be moved.

ACM is lucky to have you as an employee because it is getting quite confusing to verify if the new shuffling machine is working correctly. Your task is to write a program that, given the description of a shuffling function, predicts which slot each card will occupy after R shuffling rounds.

Input

The input contains several test cases; each one comprising two lines. The first line of each test case contains two blank-separated integers N and R, where N is the number of slots that the shuffling machine has ($1 \leq N \leq 1040$) and R is the number of shuffling rounds to be performed ($0 \leq R < 2^{63}$). The second line of each test case contains a permutation of the numbers 1, 2, …, N, separated by exactly one blank, describing the shuffling function f (the i-th number corresponds to $f(i)$, for $1 \leq i \leq N$).

The input must be read from standard input.

Output

For each test case, output a line with a blank-separated permutation of the numbers 1, 2, . . ., N, where the i-th number is the slot that the card that starts in slot i will occupy after exactly R shuffling rounds, for $1 \leq i \leq N$. Do not print a blank after the last number in each line.

The output must be written to standard output.

Sample input	**Output for the sample input**

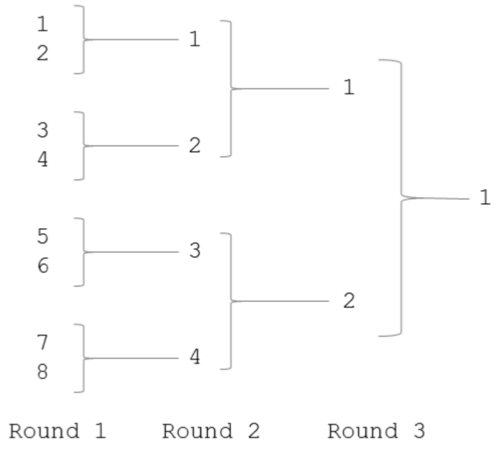
```
5 0
4 2 1 5 3
5 1
4 2 1 5 3
5 2
4 2 1 5 3
```
	```
	1 2 3 4 5
	4 2 1 5 3
	5 2 4 3 1
	```

Problem G
Tennis Rounds

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

In an N-rounds tennis tournament a group of 2^N players is seeded defining the first round to be played. Seeding means that each player is assigned a number between 1 and 2^N, and this assignment defines the round's draw as it establishes the matches that will be played among the players. In particular, the first round matches are numbered 1, 2, …, 2^{N-1}: match k will have player $2{\cdot}k{-}1$ vs. player $2{\cdot}k$, for $1{\le}k{\le}2^{N-1}$.

The winner of a match in the first round advances to the second round and the loser is eliminated. Consequently, the second round has exactly half the players of the first round. Moreover, if the winner of the first round match k would be reassigned the number k, then the second round's draw may be defined exactly as already explained for the first round. This assignment process could be repeated over and over again until there is exactly one player remaining, who happens to be the tournament's champion:



It is clear that the seeding process and the subsequent draws make it possible for any two players to eventually face each other in some round. For example, for N=3, players 2 and 5 could play at round 3 (the final), and players 5 and 7 could play at round 2 (one of the semifinals).

The tennis tournament organization is developing an online portal featuring many services. You have been hired to implement one of such services: given the seeding numbers of two players (i.e., their place in the first round ordering), the service should compute the round number in which these two players could eventually have a match against other.

Input

The input consists of several test cases, each one defined by a line containing three blank-separated integers N, i, and j, where N indicates the total number of rounds in the tournament ($1{\le}N{\le}20$), and i, j represent two seeding numbers at the first round ($1{\le}i{\le}2^N$, $1{\le}j{\le}2^N$, $i{\ne}j$).

The input must be read from standard input.

Output

For each test case, output a line with one integer indicating the round number in which players i and j may have a match in a tournament with N rounds.

The output must be written to standard output.

Sample input	**Output for the sample input**
`3 2 5`	`3`
`3 5 7`	`2`
`2 1 2`	`1`
`2 2 1`	`1`

Problem H
Vocabulary

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

Jack and Jill were arguing about which one of them had the richest vocabulary. Because neither one of them gave up, Jill proposed a game to test Jack's vocabulary:

- Jill would write down a list of strings, each one of them called a *challenge*.
- A challenge s in Jill's list is *solved* by Jack if he could say a word w from his vocabulary, such that every letter occurs in w with at least the same multiplicity that occurs in s. Two different challenges may be not solved with the same word.

Jack's vocabulary is defined as the set of words he knows. Given a list of Jill's challenges, Jack would like to know how many of them he can solve.

Input

There are several cases to solve. Each case begins with a line containing blank-separated numbers V and C: the former is the number of words in Jack's vocabulary ($1 \leq V \leq 500$) and the latter is the number of challenges presented by Jill ($1 \leq C \leq 500$). Each one of the next V lines contains a word in Jack's vocabulary. Then, each one of the next C lines contains a challenge made by Jill. Both, the words in Jack's vocabulary and the challenges made by Jill, consist of 1 to 30 lowercase English letters (`a`...`z`). Jack's vocabulary does not contain duplicate words.

The input must be read from standard input.

Output

For each test case, print the maximum number of challenges that Jack can solve.

The output must be written to standard output.

Sample input	Output for the sample input
1 1 icpc pic 4 3 sequoia march may tree ae aeiou acm	1 2

Problem I
Water Supply

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

Technopark is a huge industrial park surrounded by small residential towns, inhabited by its workers. Technopark owns a dam, which provides electricity and clean drinking water to its facilities and some residential towns. The remaining towns get their water from dug wells.

Recently, some workers and their families have gotten a disease caused by bacteria found in some wells, so it has been decided to extend the water supply network to take dam water from Technopark to every residential town. Some pipelines connecting pairs of towns already exist but additional pipelines could possibly be needed. Pipelines are unidirectional, i.e., they only allow water transportation in one direction, and every pipeline connects exactly two towns.

Your task is to compute the minimum number of pipelines that must be installed to take dam water to every residential town.

Input

The input contains several test cases. The first line of each test case has two blank-separated integers N and M, where N is the number of residential towns ($1 \leq N \leq 1000$) and M is the number of existing pipelines ($0 \leq M \leq 100000$). Towns are numbered from 0 to N, being Technopark town 0. Each of the next M lines contains two blank-separated integers a and b ($0 \leq a \leq N$, $0 \leq b \leq N$) indicating that there is a pipeline taking water from town a to town b.

The input must be read from standard input.

Output

For each test case, print the minimum number of pipelines that must be built to take dam water to every residential town.

The output must be written to standard output.

Sample input	**Output for the sample input**

4 5
0 1
1 2
2 1
0 4
3 4
4 2
3 1
2 1
	1
	3