Mock CCC 2014 by Alex and Timothy Li (Senior Division)

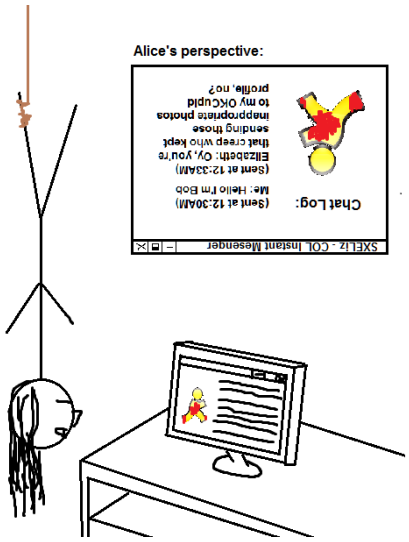
These problems are available for online grading on http://wcipeg.com/contests, however, due to latency issues with the server, they have been released by PDF as an alternative. If you would like your solutions to be graded before the CCC (February 25, 2014), either wait for our online judge to respond, or feel free to email your solutions to wcipeg@gmail.com (attach your source files to the same email message, naming it something clear like "s1.cpp", "s2.java", etc.). We'll be glad to grade it for you ASAP and reply with your score and grading details before the real contest. Analyses, test data, and solutions will be released Sunday night (February 23rd, 2014).

In this contest, there are 5 problems of increasing difficulty, each scored out of 15, for a total of 75 points. You should give yourself 3 hours to complete the problems. All I/O are from standard input and output. Do not read from any files. Remember that on the real CCC, machine-readable materials (e.g. forums, programs you have written previously.) are not allowed, but language reference (such as www.cplusplus.com or http://docs.oracle.com/javase/7/docs/api/), as well as books and printed-materials are allowed.

We've put a lot of work into these problems, and we hope you enjoy them.

Good luck!

Problem S1: Reverse Find



Alice has an intense, but secret crush on her friend Bob, though she's pretty sure it's not mutual. She would like to learn about his interactions with other girls that may end up being her competition. After paying for some spy training, she managed to find her way into Bob's apartment. Bob had just left to pick up some errands, and he happened to have left his instant messaging program open. Alice is now hanging on a rope from Bob's ceiling in front of his computer, staring directly into a chat log of all of his conversations.

She would like to bookmark the location of some keywords in the chatlog so she can look them up later when she'll have obtained the entire file. The only problem is, because she's hanging upside-down, the text is reversed and upside-down from her eyes! Luckily, she can read letters upside down, but she'll need your help bookmarking the keyword in reverse.

Specifically, given a reversed text *T* of no more than 1000 ASCII characters, and a keyword pattern *P* of no more than 100 ASCII characters, Alice needs to find the index of the first occurrence of the pattern when the string is read in reverse.

Input Format

Line 1 of input contains the text string *T*. Line 2 of input contains the pattern string *P*.

Output Format

Output 1 integer — the index of the first occurrence of the pattern when the string is read in reverse. Comparisons of the text and pattern are case-sensitive. The first character of input is index 1, the second character is index 2, and so on. If the pattern cannot be reverse-found, output -1.

Sample Input 1

```
.boB m'I olleH :eM (MA03:21 ta tneS)
Bob
```

Sample Output 1

```
4
```

Sample Input 2

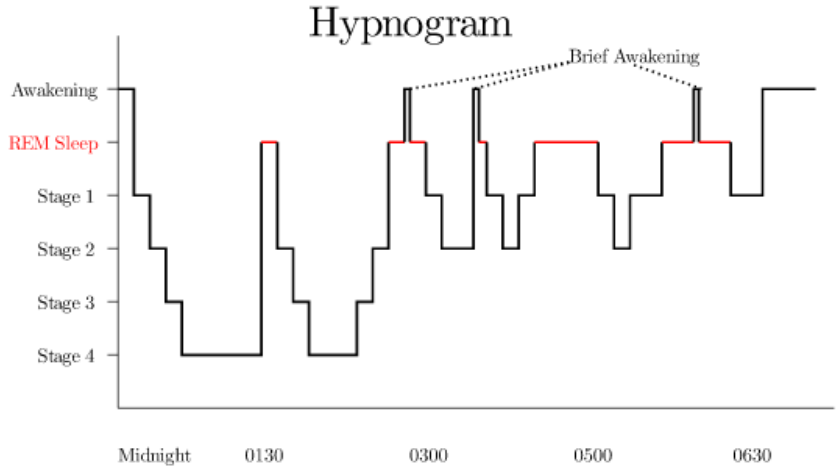
```
.tah eht ni tac eht :eM (MA04:21 ta tneS)
the
```

Sample Output 2

```
19
```

Problem S2: Sleep Cycle

Alice's obsession with her friend Bob is so tenacious that she cannot fall asleep at night. She decided to come up with a plan to secretly visit Bob in his house while he is sleeping. To not get caught, she must go only when she knows that Bob will be sound asleep for a while. Alice knows that people have sleep cycles. Below is an example of a hynogram, demonstrating the human sleep cycle.



However, every individual's sleep pattern is different and can be affected by many different factors. For instance, Bob's sleep cycle can be different due to his habit of staying up late to watch movies, or due to him being a heavier sleeper than most. The only thing Alice can truly rely on is the cold, hard, data she has collected from a high-quality microphone, previously planted outside of Bob's window to monitor his sleeping patterns.

Given N ($3 \leq N \leq 1000$) measurements of Bob's awakeness level recorded at fixed time intervals, Alice would like to know the length of the longest *dip* in the data. A dip is a consecutive sequence of length three or more that starts at a value, progresses to the minimum value of the sequence in a strictly decreasing pattern, remains at the minimum value for one or more iterations, and progresses back up in a strictly increasing pattern. More specifically, a consecutive subsequence from i to j of a sequence of numbers is a dip if and only if $x_i > x_{i+1} > \ldots > x_k$ ($= x_{k+1} = x_{k+2} = \ldots$) $< \ldots < x_{j-1} < x_j$. For example, the sequence {7, 5, 4, 2, 1, 2, 8, 10} is a dip of length 8, the sequence {3, 2, 2, 3} is a dip of length 4, but the sequences {1, 1, 1}, {5, 4, 4, 2, 3, 5} and {5, 2, 1, 9, 8, 10} are not dips. Knowing the length of the longest dip, Alice will then be able to determine whether she has an opportunity to break in and watch Bob sleep.

Input Format

Line 1 contains the positive integer N, the number of data points to follow.
Line 2 contains N positive integers each no greater than 1 billion, the measurements of Bob's awakeness levels that Alice has recorded.

Output Format

One integer — the length of the longest dip, or 0 if the input does not contain a dip.

Sample Input

```
12
21 5 4 10 8 4 3 6 12 17 4 17
```

Sample Output

```
7
```

Explanation

The longest dip in the data is 10, 8, 4, 3, 6, 12, 17.

Problem S3: Spreadsheet Sorting

Over the years, Alice has collected the physical data of her friend and secret love interest, Bob. Now she's trying to find trends in the data so she can learn more about Bob, and thus get closer to him. Alice entered the data into her spreadsheet software, but learned that unfortunately, the software doesn't work as it should. Namely, the "sort by column" function is broken. Each time Alice clicks a column, she wants all the rows in the spreadsheet to be sorted in nondecreasing order by values from that column. If rows have the same values in the column she clicks, the *relative order* of these rows should be preserved after the sort. Given the original spreadsheet and the sequence of the columns that Alice clicked, you are to determine the sorted spreadsheet.

Input Format

Line 1 of input contains two integers: R, the number of rows in the spreadsheet, and C, the number of columns in the spreadsheet ($1 \leq R, C \leq 100$).
The next R lines will each contain C integers between 1 and 10 000 inclusive, representing single cells of the spreadsheet.
The next line of input will have an integer N ($1 \leq N \leq 100$), the number of sorts that Alice makes.
The last N lines will each contain a number c_i ($1 \leq c_i \leq C$), indicating that she sorts by column c_i for the i-th command ($1 \leq i \leq N$).

Output Format

The output should contain R lines with C integers per line, the sorted spreadsheet after the N clicks.

Sample Input

```
4 3
6 2 1
9 1 3
9 2 1
6 1 1
2
2
1
```

Sample Output

```
6 1 1
6 2 1
9 1 3
9 2 1
```

Explanation

Alice first sorts by column 2, then she sorts by column 1. The spreadsheet is sorted as follows:

```
Before sorting:    Sort by col 2:     Sort by col 1:
    6 2 1              9 1 3              6 1 1
    9 1 3     ---->    6 1 1     ---->    6 2 1
    9 2 1              6 2 1              9 1 3
    6 1 1              9 2 1              9 2 1
```

Problem S4: Roadtrip Tracking

Alice heard that her secret love interest, Bob, is going on a roadtrip to visit his friend in another city. She decides to follow him along the way to see if the friend is actually romantically affiliated with Bob (in which case, the outcome is not going to be pretty). Alice does not know the route that Bob is going to take, so she has acquired a map of the country. In their country, there are N ($1 \leq N \leq 40\,000$) cities conveniently labeled from 1 to N, and M ($1 \leq M \leq 200\,000$) bidirectional roads that run between the cities. There is at most one road between any pair of cities.

Both of them are currently in city 1, and Bob would like to go to city N. Alice also wants to get to city N eventually, but how she gets there does not matter. Bob's route will never return to the same city twice, and Alice does not plan to waste time by returning to the same city in her route either. The problem is, Alice cannot take the *exact* same route as Bob because Bob might get suspicious of a car that's following him for the entire trip. At some point, Alice will have to split up with Bob for a while. Alice needs to know whether there are at least two distinct ways to get from city 1 to city N.

Scoring

In addition to the constraints above, the following will hold:

- For test cases worth 5/15 of the points, $N \leq 10$, $M \leq 20$.
- For test cases worth 10/15 of the points, $N \leq 1\,000$, $M \leq 10\,000$.

Input Format

Line 1 contains the two integers N and M, denoting the number of cities and roads in their country.
The next M lines each contain two integers, a and b, indicating that there is a two-way road between city a and city b.

Output Format

Output "`Yes`" if there are at least two different ways to get from city 1 to city N without revisiting the same city twice, otherwise "`No`".

Sample Input

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

Sample Output

```
Yes
```

Explanation

Under the conditions above, there are exactly two ways to get from city 1 to city 6 — either by taking the path 1→2→3→5→6 or the path 1→2→4→5→6.

Problem S5: Elimination Game

Alice had recently pursued her (not so) secret love interest, Bob, on a long roadtrip to see just who he was meeting in another town. To her dismay, she learned that Bob was getting back together with one of his ex-girlfriends. As she made the discovery, her body convulsed with rage, the sky spun around her head, and the ground trembled under her feet. One thing led to another; before you know it, Bob has found himself tied up and gagged in Alice's basement. Because Bob is a computer scientist, Alice stands before him offering him only two options — play the *Elimination Game*with her, or prepare to be eliminated. If she can't have him, nobody can!

To play the *Elimination Game*, Alice presents Bob with two piles of cards — a pile of N black cards and a pile of M white cards ($1 \leq N, M \leq 100$). Every card has a positive integer no greater than 100 000 written on it. The rules of the game are as follows:

- Bob picks two cards — one from the black pile and one from the white pile, such that the numbers on both cards share a factor greater than 1.
- The two cards are eliminated from their respective piles.
- Bob repeats this process until there are no more pairs that can be eliminated.
- The objective is to eliminate as many cards as possible.

Using many days and nights of computational power from a supercomputer, Alice has obtained the maximum (total) number of cards that can be eliminated from the two piles under these rules. If Bob finds this number, then he is free to go. Otherwise, he himself will be forever "eliminated" by the crazed, lust-driven Alice. Write a program that helps Bob escape!

Scoring

Bob immediately comes up with a strategy. He decides to start at the beginning of the black cards and go through them one by one. For each card in the black pile, he goes through all of the (non-eliminated) cards in the white pile from beginning to end. If at any one point the two cards he's examining can be eliminated, he eliminates them and moves on to the next black card. If no white cards share a factor greater than 1 with the black card, he then skips to the next black card. Clearly, this strategy is not perfect and will not always lead to the correct answer. However, implementing it efficiently will get you at least 5/15 of the points.

On an unrelated note:

- For test cases worth 5/15 of the points: $N, M \leq 5$.
- For test cases worth 10/15 of the points: $N, M \leq 10$.

Input Format

Line 1 of input contains the two integers N and M.
Line 2 contains N integers, the values of the black cards.
Line 3 contains M integers, the values of the white cards.

Output Format

The output should contain one integer, the maximum total number of cards that can be eliminated from the piles.

Sample Input

```
5 5
3 10 6 7 17
3 5 14 4 15
```

Sample Output

```
8
```

Explanation

One way to elimate 8 of the 10 cards is to eliminate the pairs (3, 3), (10, 5), (6, 4), and (7, 14), with 17 left over in the black pile and 15 left over in the white pile. Another way is to eliminate the pairs (3, 15), (10, 5), (6, 3), and (7, 14), with 17 left over from the black pile and 4 left over from the white pile. We know that it's not possible to eliminate all 10 cards because we cannot eliminate 17, since it does not share factors greater than 1 with any other number except for itself (and there are no 17's in the white pile). Therefore, 8 is the most we can eliminate.