2023 ACM Programming Contest

- Complete as many questions as you can during the contest.
- **Only one computer may be used per team.**
- No connection to the Internet may be made during the contest.
- All questions require you to read the test data from standard input and write results to standard output; you cannot use any non-contest files for input or output.
- The input to problems will consist of multiple test cases unless otherwise noted.
- Programming style is not considered in this contest. You are free to code in whatever style you prefer. Commenting code is not required.
- All communication with the judges will be handled by the PC2 environment.
- Allowed programming languages: C, C++, Java, and Python.
- **Java users: remember to call `nextLine()` on a scanner after reading in an integer when needed.**
- All programs will be re-compiled prior to testing with the judges' data.
- Output must match the sample format exactly. Do not print prompts unless asked to.
- Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and string libraries are allowed. The standard Java API is available and can be used, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation)
- Hardcopy reference materials are allowed, but not electronic material. Calculators are fine, but no laptops, phones, USB drives, etc. are allowed.
- **Netbeans users: Remove package statement before you submit**

Judges decisions are to be considered final. No cheating will be tolerated.

Problem A - Every Second Counts

Meredith runs a taxi service called Ruber which offers rides to clients in small towns in western Pennsylvania. She wants to get every possible dime out of people who use her taxis, so her drivers charge a flat fee not per minute but per second. It's important, therefore, to be able to determine the exact amount of elapsed time between the moment a client enters a cab until the moment they leave. Trying to write a program to do this has driven Meredith crazy (pun intended) so she's come to you for some help.

Input Input consists of two lines: the first contains the start time and the second contains the end time for a single taxi ride. Each time is of the form $hh : mm : ss$, giving the hour, minute and seconds. Meredith uses a 24 hour clock, with $0 : 0 : 0$ representing 12 midnight and $23 : 59 : 59$ representing one second before midnight. Note that the end time may have a value less than the start time value if the ride spans midnight (see the last sample test case for an example of this).

Output Display the number of seconds between the two times. No cab ride will be equal to or longer than 24 hours.

Sample Input 1

```
10 : 0 : 0
11 : 0 : 0
```

Sample Output 1

```
3600
```

Sample Input 2

```
23 : 0 : 0
1 : 30 : 0
```

Sample Output 2

```
9000
```

Problem B - What Does the Fox Say?

Determined to discover the ancient mystery - the sound that the fox makes - you went into the forest, armed with a very good digital audio recorder. The forest is, however, full of animals' voices, and on your recording, many different sounds can be heard. But you are well prepared for your task: you know exactly all the sounds which other animals make. Therefore the rest of the recording - all the unidentified noises - must have been made by the fox.

Input The first line of each test case contains the recording - words, separated by spaces. Each contains at most 100 letters and there are no more than 100 words. The next few lines are your pre-gathered information about other animals, in the format `<animal> goes <sound>`. There are no more than 100 animals, their names are not longer than 100 letters each and are actual names of animals in English. There is no fox goes ... among these lines. The last line of the test case is exactly the question you are supposed to answer: what does the fox say?

Output Output one line containing the sounds made by the fox, in the order from the recording. You may assume that the fox was not silent (contrary to popular belief, foxes do not communicate by Morse code).

Sample Input

```
toot woof wa ow ow ow pa blub blub pa toot pa blub pa pa ow pow toot
dog goes woof
fish goes blub
elephant goes toot
seal goes ow
what does the fox say?
```

Sample Output

```
wa pa pa pa pa pa pow
```

Problem C - Calorie Counting

Stan Ford is a typical college graduate student, meaning that one of the most important things on his mind is where his next meal will be. Fortune has smiled on him as he's been invited to a multi-course barbecue put on by some of the corporate sponsors of his research team, where each course lasts exactly one hour. Stan is a bit of an analytical type and has determined that his eating pattern over a set of consecutive hours is always very consistent. In the first hour, he can eat up to m calories (where m depends on factors such as stress, bio-rhythms, position of the planets, etc.), but that amount goes down by a factor of two-thirds each consecutive hour afterwards (always truncating in cases of fractions of a calorie). However, if he stops eating for one hour, the next hour he can eat at the same rate as he did before he stopped. So, for example, if $m = 900$ and he ate for five consecutive hours, the most he could eat each of those hours would be 900, 600, 400, 266 and 177 calories, respectively. If, however, he didn't eat in the third hour, he could then eat 900, 600, 0, 600 and 400 calories in each of those hours. Furthermore, if Stan can refrain from eating for two hours, then the hour after that he's capable of eating m calories again. In the example above, if Stan didn't eat during the third and fourth hours, then he could consume 900, 600, 0, 0 and 900 calories.

Stan is waiting to hear what will be served each hour of the barbecue as he realizes that the menu will determine when and how often he should refrain from eating. For example, if the barbecue lasts 5 hours and the courses served each hour have calories 800, 700, 400, 300, 200 then the best strategy when $m = 900$ is to eat every hour for a total consumption of $800 + 600 + 400 + 266 + 177 = 2,243$ calories. If however, the third course is reduced from 400 calories to 40 calories (some low-calorie celery dish), then the best strategy is to not eat during the third hour — this results in a total consumption of $1,900$ calories.

The prospect of all this upcoming food has got Stan so frazzled he can't think straight. Given the number of courses and the number of calories for each course, can you determine the maximum amount of calories Stan can eat?

Input Input starts with a line containing two positive integers n and m ($n \leq 100, m \leq 20,000$) indicating the number of courses and the number of calories Stan can eat in the first hour, respectively. The next line contains n positive integers indicating the number of calories for each course.

Output Display the maximum number of calories Stan can consume.

Sample Input

```
5 900
800 700 400 300 200
```

Sample Output

```
2243
```

Problem D - Tweet Tweaks

Ken has been having trouble lately staying under the word limit in Twitter, so he's decided to write a little front-end program which will take in text and shorten it using a fixed set of abbreviations for commonly used letter sequences. Those abbreviations are shown in the table below:

character	substitutes for
@	at
&	and
1	one, won
2	to, too, two
4	for, four
b	bea, be, bee
c	sea, see
i	eye
o	oh, owe
r	are
u	you
y	why

Ken is about to start writing this program when he realizes that the extent of his computer knowledge is, well, using Twitter. He's looking for someone to help him – r u th@ some1?

Input Input starts with a single integer n indicating the number of lines of text to process. Following this are n lines of text. Each line will contain only alphanumeric characters and spaces, and each line will have at least one non-space character. Each line has at most 200 characters.

Output Display each line with the appropriate substitutions made. Substitutions should also be made inside words, e.g., the word *that* should be changed to *th@*. If two letter sequences overlap (like *at* and *to* in the word *baton*) just replace the first one (in this case resulting in *b@on*). If two letter sequences start at the same location (like *be* and *bee* in *been*) replace the longer one (in this case resulting in *bn*). If the letter sequence starts with an upper-case letter, then the abbreviation should also be in upper-case (if appropriate). Finally, no substituted letter should later be part of another substitution. For example, if the input is *oweh*, you would first replace the *owe* with an *o* to get *oh*. At this point you do NOT replace the *oh* with an *o* since the *oh* contains a substituted letter.

Sample Input

```
3
Oh say can you see
I do not understand why you are so cranky just because Karel won
Formation
```

Sample Output

```
O say can u c
I do not underst& y u r so cranky just bcause Krl 1
4m@ion
```

Problem E - A Random Problem

Generating a random number sequence is not easy. Many sequences may look random but upon closer inspection we can find hidden regularities in the occurrence of the numbers. For example, consider the following 100-digit "random" sequence starting with `4 7 9...` :

```
4 7 9 5 9 3 5 0 0 1 7 8 5 0 2 6 3 5 4 4
4 6 3 3 2 7 1 8 7 8 7 6 1 1 7 2 5 4 7 2
0 4 4 5 8 3 0 6 9 3 2 6 6 8 5 2 5 1 2 7
2 4 1 0 0 4 9 1 8 7 5 0 4 4 8 4 3 2 6 8
8 5 6 7 0 9 7 0 3 6 1 4 4 1 2 3 2 6 9 9
```

If you look closely, whenever a 4 is followed immediately by another 4, the third value after this will be a 3 (we're sure you saw that). We'll call features like this *triple correlations* and we'll represent it as 4(1)4(3)3, where the numbers in parentheses indicate how far away each number on either side of the parentheses must be. More precisely, a sequence of p digits has an $a(n)b(m)c$ triple correlation if

1. any time there is an a followed n locations later by a b there is always a c located m locations after the b unless b's location is within distance $m-1$ of the end of the sequence.
2. any time there is a b followed m locations later by a c there is always an a located n locations before the b unless b's location is within distance $n-1$ of the beginning of the sequence.
3. any time there is an a followed $n+m$ locations later by a c there is always a b located n locations after the a.
4. the correlation occurs at least $\lceil p/40 \rceil + 1$ times, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Such correlations are tough to spot in general, so that's where we need your help. You will be given a sequence of digits and must search it for triple correlations.

Input Input starts with a positive integer p ($p \leq 1\,000$) indicating the number of random digits in the sequence. Starting on the next line will be the p digits, separated by spaces and potentially spread across multiple lines. No line will contain more than 100 digits and there will be no blank lines.

Output Display `triple correlation a(n)b(m)c found` if it occurs in the list (with the appropriate values for `a`, `b`, `c`, `n`, and `m` filled in) or the phrase `random sequence` otherwise.

If there are multiple triple correlations in the sequence, display the one which has its first occurrence (as determined by the first occurrence of its a-value) earliest in the sequence. If there is still a tie, choose the one with the smaller value of n, and if there is still a tie, choose the one with the smaller value of m.

Sample Input 1

```
100
4 7 9 5 9 3 5 0 0 1 7 8 5 0 2 6 3 5 4 4
4 6 3 3 2 7 1 8 7 8 7 6 1 1 7 2 5 4 7 2
0 4 4 5 8 3 0 6 9 3 2 6 6 8 5 2 5 1 2 7
2 4 1 0 0 4 9 1 8 7 5 0 4 4 8 4 3 2 6 8
8 5 6 7 0 9 7 0 3 6 1 4 4 1 2 3 2 6 9 9
```

Sample Output 1

```
triple correlation 4(1)4(3)3 found
```

Sample Input 2

```
10
1 2 3 1 2 2 1 1 3 0
```

Sample Output 2

```
random sequence
```

Problem F - Prime Plates

Meredith has just bought a new car and needs license plates. In Pennsylvania, license plates consist of three uppercase letters followed by four digits. License plates are given out in lexicographic order, meaning that plate AAA 0000 is first, followed by AAA 0001, AAA 0002, etc,. After AAA 9999 comes plates AAB 0000, AAB 0001, etc., and after plate AZZ 9999 comes plate BAA 0000. The very last plate is ZZZ 9999.

For most people, any old license plate will do, but not for Meredith. She insists that the 4 digit number on the plate must be a prime number. Recall that a prime number is a number > 1 whose only factors are 1 and the number itself. The first few prime numbers are:

$2, 3, 5, 7, 11, 13, 17, 19, 23, ...$

So if Meredith goes to the DMV to get a set of plates, and the next available plate is JPB 1285, she'll wait until plate JPB 1289 becomes available, since 1289 is the first prime greater than or equal to 1285. If the next available plate had been JPB 9999, she would have waited for the plate JPC 0002.

Perhaps you see what we're driving at here. Starting at a given license plate, find the first plate lexicographically equal or greater that contains a prime number.

Input The input file contains multiple test cases. Each test case is on a single line and contains a three character string followed by a 4 digit number. The three character string consists solely of uppercase letters other than the string "ZZZ". A line containing END 0000 terminates input.

Output For each test case, display the license plate Meredith will accept.

Sample Input

```
JPB 1285
JPB 9999
END 0000
```

Sample Output

```
JPB 1289
JPC 0002
```

Problem G - Zoo Trotter

You are an employee at the local zoo and have recently been promoted from excretory extraction to overseeing the layout of all the sidewalks throughout the property. Currently all the sidewalks are 1-way, and the zoo is set up in various *zones*. Each zone is made up of a set of *attractions* (elephant enclosure, lizard house, etc.), and within any one zone you can get from any attraction to any other attraction in the zone using one or more sidewalks. Once you take a sidewalk from one zone to another you can never return to the zone you left. However, it is possible to walk to every zone in a single visit to the zoo. The original designers thought this arrangement was very important to help control the flow of visitors to the zoo.

The members of the board of directors have come to you with a problem. They agree with the original designers in the use of zones, but they feel that more one-way sidewalks could be included to make the zoo a little more patron-friendly. They would like you to figure out the maximum number of sidewalks that could be added so as not to introduce a path between any two attractions which didn't have one between them before.

For example, consider the small zoo shown in Figure G.1, with 7 attractions labeled 1 (the "Camel Castle") through 7 (the "Hippo Hippodrome"). Currently attractions 1, 2, 3 and 4 form one zone and 5, 6 and 7 form another. You can add sidewalks from 1, 2, 3 or 4 to either 5, 6 or 7, but adding a sidewalk from (say) 7 to 1 would allow patrons to get from 7 to 1, which was impossible previously. Note that you can also add sidewalks between all attractions within a zone which don't already exist (for example, from 1 to 3 or from 2 to 4). The total number of possible new sidewalks for this zoo would be 21.

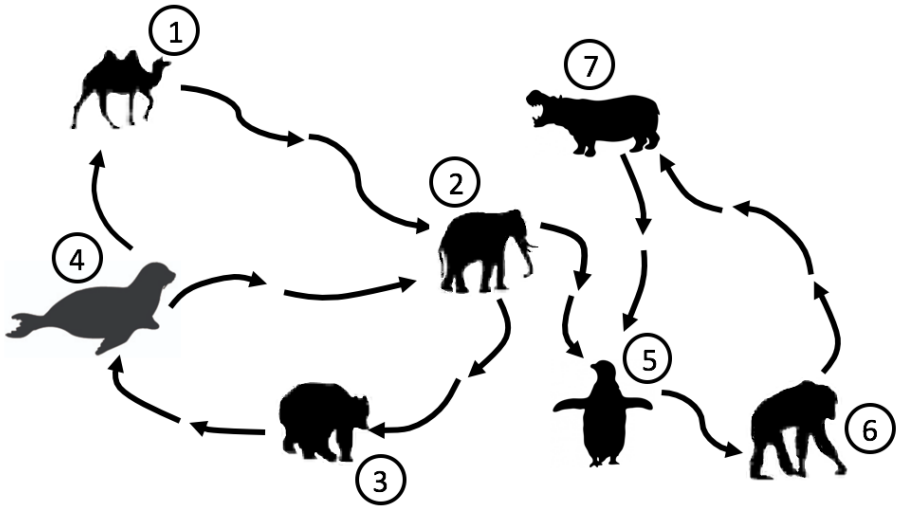

Figure G.1: Sample zoo. This example corresponds to Sample Input 1.

Input Input starts with a line containing an integer n where $(1 \leq n \leq 2,500)$, the number of attractions, labeled 1 to n. After this are n lines each containing n integers. If the j-th integer on the i-th of these lines is a 1 it indicates that there is a one-way sidewalk from attraction i to attraction j; otherwise this integer will be a 0 indicating no such sidewalk is present. There is never a sidewalk from an attraction to itself.

Output Display the maximum number of new one-way sidewalks that could be added to the zoo.

Sample Input

```
7
0 1 0 0 0 0 0
0 0 1 0 1 0 0
1 0 0 1 0 0 0
1 0 0 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 1
0 0 0 0 1 0 0
```

Sample Output

```
21
```

Problem H - The Key to Cryptography

Suppose you need to encrypt a top secret message like: `SEND MORE MONKEYS` You could use a simple substitution cipher, where each letter in the alphabet is replaced with a different letter. However, these ciphers are easily broken by using the fact that certain letters of the alphabet (like `E`, `S`, and `A`) appear more frequently than others (like `Q`, `Z`, and `X`). A better encryption scheme would vary the substitutions used for each letter. One such system is the autokey cipher.

To encrypt a message, you first select a secret word such as `ACM` and prepend it to the front of the message. This longer string is truncated to the length of the message and called the key, and the n-th letter of the key is used to encrypt the n-th letter of the original message.

This encryption is done by treating each letter in the key as a cyclic shift value for the corresponding letter in the message, where `A` indicates a shift of 0, `B` a shift of 1, and so on. Using `ACM` as the secret word, we would encrypt our message as follows:

Data	Cryptographic element
`SENDMOREMONKEYS`	Message
`ACMSENDMOREMONK`	Key
`SGZVQBUQAFRWSLC`	Ciphertext

Note that the letter 'E' in the message was encrypted as 'G' the first time it was encountered (since the corresponding letter in the key was 'C' indicating a shift of 2), but then as 'Q' and 'S' the next two times.

Your task is simple: given a ciphertext and the secret word, you must determine the original message.

Sample Input

```
SGZVQBUQAFRWSLC
ACM
```

Sample Output

```
SENDMOREMONKEYS
```

Problem I - Taxed Editor

Reed Dacht is a freelance book editor who works out of his home. Various publishers send him texts to read along with a deadline date for when they would like his editorial comments to be sent back to them. Reed is very precise with his reading and will never read more than one book at a time, so he never starts a new book until after he is finished reading the previous one. He is willing to change the rate (number of pages per day) at which he reads books, but to be fair to each of his clients he insists on using the same speed for all books.

The other day Reed got a large set of editing requests and has a bit of a problem. He doesn't think that there's any way he can read all of the books by the specified deadlines. He could set his reading speed high enough to get them all done in time, but that might compromise the accuracy of his editing (as well as severely tax his eyesight). To avoid that he could select a slower reading speed, but then too many of the editing jobs would go past their deadlines. He has decided on a compromise: he is willing to let a small number of books go past deadline (so as to annoy only a small number of clients) and will set his reading speed to the smallest possible number of pages per day so that no more than that number of books will be late. But what should that reading speed be? That's where you come in.

Input Input starts with two integers n and m where n ($1 \leq n \leq 10^5$) is the number of books to edit and m ($0 \leq m < n$) is the number of editing jobs that Reed will allow to be late.

Following this are n lines each describing one book. Each line contains two integers l and d where l ($1 \leq l \leq 10^9$) is the length (in pages) of the book and d ($1 \leq d \leq 10^4$) is the deadline for that book (number of days until is it due).

Output Output the minimum integer speed (pages per day) which allows no more than m late editing jobs.

Sample Input

```
3 1
450 9
500 6
300 4
```

Sample Output

```
84
```

Problem J - Sheba's Amoebas

After a successful Kickstarter campaign, Sheba Arriba has raised enough money for her mail-order biology supply company. "Sheba's Amoebas" can ship Petri dishes already populated with a colony of those tiny one-celled organisms. However, Sheba needs to be able to verify the number of amoebas her company sends out. For each dish she has a black-and-white image that has been pre-processed to show each amoeba as a simple closed loop of black pixels. (A loop is a minimal set of black pixels in which each pixel is adjacent to exactly two other pixels in the set; adjacent means sharing an edge or corner of a pixel.) All black pixels in the image belong to some loop.

Sheba would like you to write a program that counts the closed loops in a rectangular array of black and white pixels. No two closed loops in the image touch or overlap. One particularly nasty species of cannibalistic amoeba is known to surround and engulf its neighbors; consequently there may be amoebas within amoebas.

For instance, each of the images in Figure J.1 contains four amoebas.

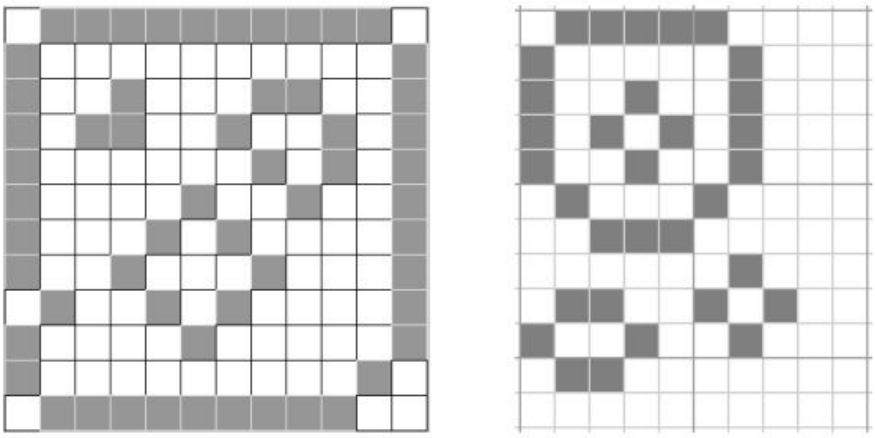

Figure J.1: Two Petri dishes, each with four amoebas.

Input The first line of input contains two integers m and n, $(1 \leq m, n \leq 100)$. This is followed by m lines, each containing n characters. A '#' denotes a black pixel, a '.' denotes a white pixel. For every black pixel, exactly two of its eight neighbors are also black.

Output Display a single integer representing the number of loops in the input.

Sample Input

```
12 12
.##########.
#..........#
#..#...##..#
#.##..#..#.#
#......#.#.#
#....#..#..#
#...#.#....#
#..#...#...#
.#..#.#....#
#....#.....#
#.........#.
.#########..
```

Sample Output

```
4
```

Problem K - Guardians

It's getting close to March Madness time and the big game this week is the classic rivalry between State U and U State. The coach of State U has been working night and day on the best strategy to beat his arch rival. On defense he has decided to have his tallest player guard the opponent's tallest player, his second tallest player guard the opponent's second tallest player, and so on. He's a bit worried about the size differences between the two teams though, and needs to know for this defensive strategy how many of his players are taller than the person they're guarding. He's asked you to take time from writing term papers for his players to write a program to calculate this value for him.

Input Input consists of two lines. The first line contains five positive integers specifying the heights of the five players on State U's team (in no particular order). The second line contains the similar information for the five players on U State's team (again in no particular order).

Output Output the number of State U players taller than the person they are guarding assuming that the tallest State U player guards the tallest U State player, the second tallest State U player guards the second tallest U State player, and so on.

Sample Input

```
67 64 61 70 58
62 68 65 71 59
```

Sample Output

```
0
```

Problem L - Hardware Sales

Bree Kim Orter owns three hardware stores in different parts of the state. She's interested in knowing what items are popular across the state, specifically those items that have sold 20 or more units in each of her stores. For each store she has a chronological list of purchases made during the past week. Each purchase record consists of two integers: the ID number of the item and the number of units purchased. An example of such a list is shown below:

```
178293 3 457839 10 322228 1 ...
```

This list indicates that the first purchase that week was three units of the item with ID #178293. The next purchase involved 10 units of item #457839, and so on. Given three of these lists (one from each store) Bree would like to determine the number of items which sold 20 or more units in each store during the week.

Input Input starts with a line containing three integers n_1, n_2, and n_3, where $(1 \leq n_1, n_2, n_3 \leq 100)$ are the number of purchases in each of the three lists.

Following this is the purchase list from the first store which contains n_1 pairs of positive integers, where the first number in each pair is a 6-digit ID number (possibly with leading zeros) and the second is the number of units sold.

These n_1 pairs may span multiple lines. After this, starting on a new line, is the purchase list of the second store and following this, again starting on a new line, is the purchase list of the third store. All purchase amounts are ≤ 100. An ID number may show up multiple times on a list.

Output Output the number of items that have sold at least 20 units in each of three stores. After this list the ID numbers of the items. Print the ID numbers in the order that they first appear in the input file.

Sample Input

```
5 2 4
123682 20 239481 23 001238 5 738299 4 001238 17
001238 31 123682 25
239481 25 123682 22 909090 18 001238 27
```

Sample Output

```
2 123682 001238
```