Problem 0. Numbers to Words

This problem involves the conversion of arbitrary integers in the range 1 to 999,999 into text. The input file for this problem has the following format.

- A first line containing a positive integer indicating how many lines follow.
- Each subsequent line contains an integer from 1 to 999,999 (without commas).

The output file will have one text line for each one of the input numbers; note that the first line of the input file is not an input number. Each such line will have a representation of the corresponding number in words (without "and" or dashes "-").

For example, the following input file

```
4
26973
45
555931
4
```

will generate the following output file

```
twenty six thousand nine hundred seventy three
forty five
five hundred fifty five thousand nine hundred thirty one
four
```

Note that the "4" on the first line is not converted.

Problem 1. Arabic to Roman Numeral conversion

In this problem you will convert a number from 1 to 3000 written with Arabic numerals (the ones we use today) into the same number written with Roman numerals.

Numbers in this range will use the following Roman numerals: I (corresponding to 1), V (5), X (10), L (50), C (100), D (500), and M (1,000). Multiple instances of I,X,C, and M can be used to represent other numbers. For example, we write 3 as III; we write 1500 as MD rather than DDD. However, it is not allowed to have more than three instances of such numerals in succession. Such cases can be resolved by having a single smaller such numeral appearing immediately before a larger numeral of the same order to indicate subtraction of the smaller from the larger numeral. Thus, an I can appear before a V or an X, but not before an L,C,D,M; an X can appear before an L or C but not before a D or M, and so on. For example, we write 4 as IV, not IIII; we write 990 as CMXC rather than XM. And of course 500 is D rather than the incorrect and weird DM.

We can convert numbers in Arabic into numbers using Roman numerals by converting, the thousands first, then the hundreds and so on skipping digits with a value of zero. Consider the example of 2904. In Roman numerals 2904 is rendered: 2000=MM, 900=CM, 4=IV; resulting in MMCMIV.

The **input file** will have the following format:

- The first line of the input file is a positive integer indicating the number of subsequent lines in the file.
- Each subsequent line contains an integer from 1 to $3,000$, inclusive.

The **output file** should contain the input numbers converted into Roman numerals, one per line.

For example, consider the following input file.

```
4
2008
448
1776
1993
```

The correct corresponding output file would be:

```
MMVIII
CDXLVIII
MDCCLXXVI
MCMXCIII
```

Problem 2. Fax Compression

A fax machine transmits a digital representation of an image. Suppose that you have a file of zeroes and ones representing a fax image, and you want to reduce the size of the file by using the following compression scheme: If there are n consecutive ones, they are replaced by n:1; similarly n consecutive zeroes are replaced by n:0. For example,

```
011000011001111110
```

would be replaced by

```
1:0 2:1 4:0 2:1 2:0 6:1 1:0
```

You are asked to implement such a fax compression method. The **input file** for this problem will have the following format:

- The first line of the file contains a positive integer (less than or equal to 32) telling you the number of subsequent lines in the file.
- Each subsequent line contains a string of 32 zeroes and ones.

The **output file** should have one line for each input line of zeroes and ones in the input file.
Every pair of n:c is separated from the next by a single space in the output.
For example, if the input is

```
2
00001011000000001111111100001111
11111011000000001111111100001111
```

then the output should be

```
4:0 1:1 1:0 2:1 8:0 8:1 4:0 4:1
5:1 1:0 2:1 8:0 8:1 4:0 4:1
```

Problem 3. Stemming

Search engines cleanup query terms before they process them. One such operation is called stemming and removes from words redundant suffixes. Suppose that you want to translate the plural form of words to singular using the following (imperfect) algorithm:

1. If the word ends in "ies" but not in "eies" or "aies" then replace the ending "ies" with "y".
2. If the word ends in "es" but not "aes", "ees" or "oes" then replace the "es" with "e".
3. If the word ends in "s" but not "us" or "ss" then remove the "s".

The **input file** will have the following format:

- The first line of the file contains a positive integer telling you the number of subsequent lines in the file.
- Each subsequent line contains a word consisting of at most twenty lower case letters $a - z$.

The **output file** should contain the singular form of each input word, computed according to the rules given above, on its own line.

For example, if the input is

```
6
bicycles
levees
horizon
separates
foxes
fuss
```

then the output should be

```
bicycle
levee
horizon
separate
foxe
fuss
```

For the first word rule number 2 was used, for "levees" rule number 3 was used, "horizon" is singular so no rule was applied, rule 2 applied to "separates" and "foxes". Note that the output "foxe" illustrates a deficiency of this elementary algorithm. (Google uses a more sophisticated rule.)

Problem 4. Anagrams

An *anagram* is a word or phrase made by transposing the letters of another word or phrase. Let's consider only words written with lower case letters and no punctuation. With this convention, some examples of anagram pairs are:

green	genre
rat	tar
rustic	citrus

Note that each letter must be used once and only once. For example, "red" and "reed" are not anagrams.

You are asked to implement an anagram-pair recognition program. The **input file** will have the following format:

- The first line of the file contains a positive integer telling you the number of subsequent lines in the file.
- Each subsequent line contains two words that are to be tested to see if they are anagrams.

The **output file** should have one line for each pair of words in the input file. If the pair of words are anagrams, the output line should contain "true", and if the words are not anagrams the line should contain "false"; quotation marks will not be printed of course.

For example, consider the input file:

```
3
pears spare
dog alligator
horse shore
```

The output file should contain

```
true
false
true
```

Problem 5. Perfect Candidate

Suppose that a committee of n people, labeled 1 to n, are trying to decide on a chairman. Each person has a subset of other members who they support for chairman. We can represent the preferences with a table of zeroes and ones, where a 1 in the ith row and jth column means that member i supports member j, and a 0 indicates that member i does not support member j. Each member wants to be chairman, and so for each i there is a one in row i, column i.

Before holding an election, the committee decides to compare their preferences and see if there is an individual who supports only himself for chairman, and who everyone else also supports; call such a member a *perfect candidate*. A perfect candidate need not exist, but it is easy to see that there can not be more than one. Write a program that takes as input a table representing preferences, and prints the index (a number between 1 and n) of a perfect candidate if one exists, and prints "none" if there is no perfect candidate.

Assume that the input is in the form: a first line containing the size n of the committee, and then n subsequent lines, each containing n integers that are either 0 or 1, separated by a space. A 1 in row i and column j indicates that member i supports member j, and a 0 indicates that i does not support j. For example, if the input is

```
4
1 1 0 1
0 1 0 0
0 1 1 1
1 1 1 1
```

then your program should print "2", since all four candidates support the second candidate, while the second candidate supports only himself.

Problem 6. Polynomial Pretty Printer

You are asked to implement a program that pretty prints a polynomial with integer coefficients. The first line of the input file for this program contains an integer n, and then $n + 1$ subsequent lines each containing an integer that is the coefficient of an n-degree polynomial (from high to low order coefficients, zero coefficients also included).

The output is a representation of the polynomial using keyboard characters that is relatively easy to read. Suppose that we want to print the polynomial $5x^2 + 2x + 1$. We would represent this with the input file

```
2
5
2
1
```

The output would be

```
5 x^2 + 2 x + 1
```

The following rules must be observed:

Rule -1 If a coefficient is -1, only the minus sign is printed.

Rule 0 A term with a coefficient of 0 is not printed.

Rule 1 An exponent or coefficient of 1 is not printed (e.g., use x instead of $1x^1$).

Rule 2 There should be no consecutive signs in the output, and no "+" on a highest order coefficient; the output `+3 x^2 + - 7 x` has two errors.

For example, for the input

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

the output should be

```
3 x^5 - x^4 + 2 x^2 + x + 6
```

Problem 7. 4 by 4 Sudoku

You are given a 4 by 4 square of digits, broken down into 4 subsquares. Each of the 16 cells contains a digit from 1 to 4, or else is blank. To solve the puzzle you place a digit in each blank cell in such a way that each row, column, and subsquare contains exactly one each of the digits from 1 to 4. For example, the solution to

	1		
		2	
			4
3			

is

2	1	4	3
4	3	2	1
1	2	3	4
3	4	1	2

The **input file** will consist of four lines, each containing 4 digits from 0 to 4, separated by blanks. A zero in the input corresponds to a blank in the puzzle. The **output file** should look like the input file, but with all the zeroes replaced with appropriate digits from 1 to 4.

For example, if the input is

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

then the correct output is

```
2 4 1 3
3 1 4 2
1 3 2 4
4 2 3 1
```

You may assume that the input represents a puzzle with a unique solution.