Important Contest Instructions!!

Please read the following instructions carefully. They contain important information on how to package and submit your solutions to the judges. If you have any questions regarding these instructions, please ask a volunteer before the start of the contest.

Program Input/Output:
Programming tasks that require interaction with the program should prompt for user input to the screen (STDOUT) and read input from the keyboard/screen (STDIN).

For non-interactive tasks, you will have two options for reading the input data. Sample data for some problems will be provided to you in a file named **probXX.txt**, where 'XX' is the problem number.

Your solution may read the input from the file probXX.txt programmatically—that is, using the File I/O constructs of your programming language.

Most problems will accept input directly from the keyboard (STDIN) instead of performing file operations. For programs with large inputs, this can become very tedious. However, an easy way to enter large amount of data is by redirecting the contents of a file to your program at runtime. For example, a file named **prob01.txt** can be redirected to STDIN of your program using syntax like this:

```
%>  java prob01 < prob01.txt
%>  java –jar js.jar prob01.js < prob01.txt
%>  python prob01.py < prob01.txt
%>  prob01.exe < prob01.txt
```

In this example you are executing prob01 and sending the contents of the file prob01.txt to the STDIN of your program. Your program should behave exactly as it would if you were typing the input at the keyboard.

> ***Tip:*** *When entering input directly from the keyboard, type 'Ctrl-Z' <return> to signal the end of input to your program.*

All program output should always be sent to the screen (STDOUT).

Submitting your Programs

Interpreted Programs (JAVA, JavaScript, Python). Your program must be named probXX.java / probXX.js / probXX.py, where 'XX' corresponds to the problem number. Please submit only the source (.java, .js or .py). For java, the main class must be named probXX. Note the capitalization. All main and supporting classes should be in the default (or anonymous) package.

Native Programs (C. C++, etc). Your program should be named probXX.exe, where 'XX' corresponds to the problem number.

You are <u>strongly</u> encouraged to submit Problem #0 (on the next page) <u>prior</u> to the start of the competition to ensure that your build environment is compatible with the judges'.

problem 0

Let the Games Begin

1 point

In the tradition of grand sporting events everywhere, such as the recent Winter Olympic Games, this year the HP CodeWars team has [virtually] constructed its own HP CodeWars Arena! Your challenges will be spread throughout the arena and the surrounding countryside. You will [virtually] travel to each event and proceed to solve each problem, gaining the posted number of points for each successful completion. The events are numbered approximately in order of increasing complexity. You do not need to solve every problem.

To get things started, you confidently, eagerly, enthusiastically, [virtually] race to the starting line. There you discover the first event is already open, awaiting your solution. For this one problem, you may submit your answer before the official contest time begins!

Input

[There is no input for this program.]

Output

Simply print the sentence below, replacing XXXX with your team number.

```
Team XXXX is ready for HP CodeWars 17!
```

problem 1
Bee Math
2 points

Your team crawls through a short tunnel into an enormous beehive and is forced to wear protective clothing to pass through a swarm of bees. At the far end of the hive, you find a formula. It reads:

The population of this bee colony can be approximated by the equation

$$P(t) = 100 * sqrt(t) + 201/(t+1) + 1$$

where t is measured in days after the colonization of the beehive.

Write a program to compute the population of the beehive after a specified number of days.

Input

Each line of input is a positive integer value for t, the number of days after the colonization of the hive. The input ends with a zero.

```
7
38
24
0
```

Output

For each value of t, the program must print t and the population of the hive for that day, rounded to the nearest integer. Your result must match the expected value within +/- 1.

```
7 291
38 623
24 499
```

problem 2
Check Digit
3 points

As you stroll by the arena gift shop, you luckily spot another event waiting for you. At the checkout register, the clerk is busy scanning UPC codes. Any product you buy has one of these. It helps identify the product being sold so that the store can manage inventory, pricing and other data. Universal Product Codes (UPC) are easy to spot by their bar code.

A UPC is a 12 digit number that encodes product information such as the manufacturer, product type, weight, and other data. The last digit in this number is the check digit. This extra digit helps verify that a tired programmer didn't get one digit wrong, or transpose a pair of numbers (ex. 34 -> 43) or otherwise alter the sequence.

Check Digit:
A check digit is used to ensure that a sequence of numbers was transmitted or entered correctly without human error. The algorithm used to calculate the check digit determines the types of errors it will catch. For UPC the algorithm catches 100% of single digit errors and 89% of transposition errors.

Your task is to calculate the missing check digit for the given list of UPCs.

Here's the UPC check digit algorithm:

- First, add all the digits in the odd-numbered positions together and multiply the result by three.
- Then, add the digits in the even-numbered positions to the result.
- Next, find modulo 10 of the sum. Modulo calculates the remainder after dividing the sum by 10.
- Finally, if the remainder is not zero, subtract it from 10.

Input
The first line of input contains the number of UPCs that follow. The digits of each UPC will be separated by one space.

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

Output
For each UPC, the program must print the UPC including the calculated check digit. The digits of each UPC should be separated by a single space.

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

problem 3
Clock Power
4 points

You amble over to the arena's scoreboard, where the event coordinators are trying to measure the power used by the clock. The clock uses four seven-segment LEDs to display the hours and the minutes. For example, as you can see in the image below, the number 1 requires lighting two segments, and the number 2 requires lighting five segments.

Lighting a single LED segment requires 15 milliamps. There is also a divider (:) between the hours and the minutes. Lighting the divider requires 20 milliamps. The first LED for the hours portion of the clock will not be lit for hour values less than 10.

Write a program to output the number of milliamps required to light the clock for a given a time value.

Input

Input begins with a single integer T (T < 50) representing the number of time values. The following T lines will contain a single time value of the format HH:MM. The HH value will be a number from 1 to 12. There will be no leading zero for hour values less than 10. The MM value will be a number from 0 to 59. The MM value will have a leading zero for values less than 10.

```
6
1:23
10:58
12:00
3:14
1:11
7:38
```

Output

For each time value, output the number of milliamps required.

```
200 milliamps
320 milliamps
305 milliamps
185 milliamps
110 milliamps
245 milliamps
```

problem 4
Candy Count
5 points

You stumble into the refreshment tent for some much needed nutrition, but you discover jars of candy instead.

On each table is one jar and a list of names and numbers. Apparently people have been making guesses about how much candy is in each jar. Now we need to know whose guesses were closest.

Write a program to determine the winner(s) with the closest guesses to the number of candies in each jar.

Input

Input for each jar begins with an integer C (C < 1000) describing the actual candy count for that jar.
The next line contains the number of people P (P < 30) guessing.

The following P lines contain the guess and name of the person. The guess is an integer G (G < 1000). A single space will separate the guess from the name. The name will consist only of uppercase and lowercase letters A-Z. Names will not contain any spaces.

Example 1:
```
480
4
90 John
400 Melinda
560 Chuck
173 Miika
```

Example 2:
```
362
5
123 Miika
456 John
321 Chuck
400 Melinda
314 David
```

Output

Output the name of the winner (the one with the closest guess for the jar.) If multiple people are just as close to the actual candy count, output their names in the order they appear in the input, separated by spaces.

Example 1:
```
Melinda Chuck
```

Example 2:
```
Melinda
```

problem 5
Pangram
5 points

You step into the next event tent, which has been decorated like a school classroom. Around the room are all the letters of the alphabet. On the large board at the front of the room, you see:

The quick brown fox jumps over the lazy dog.

The bulletin board on the wall explains: A sentence that uses every letter of the alphabet is called a pangram. A "perfect" pangram only uses each letter once.

Write a program to determine if a sentence is a pangram or a perfect pangram.

Input

Each line of input is a sentence that ends with a period. Other punctuation may also be included. The input ends with a single period.

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
ALL YOUR BASE ARE BELONG TO US; SOMEONE SET US UP THE BOMB.
"NOW FAX A QUIZ TO JACK!" MY BRAVE GHOST PLED.
QUICK HIJINX SWIFTLY REVAMPED THE GAZEBO.
NEW JOB: FIX MR GLUCK'S HAZY TV, PDQ.
LOREM IPSUM DOLOR SIT AMET CONSECTETUR ADIPISCING ELIT.
PACK MY BOX WITH FIVE DOZEN LIQUOR JUGS.
.
```

Output

For each sentence, the program must print the key word PERFECT if the sentence is a perfect pangram, PANGRAM if it is a non-perfect pangram, or NEITHER if it is neither one. After the key word, the program must also print a colon, a space, and the original sentence.

```
PANGRAM: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
NEITHER: ALL YOUR BASE ARE BELONG TO US; SOMEONE SET US UP THE BOMB.
PANGRAM: "NOW FAX A QUIZ TO JACK!" MY BRAVE GHOST PLED.
PANGRAM: QUICK HIJINX SWIFTLY REVAMPED THE GAZEBO.
PERFECT: NEW JOB: FIX MR GLUCK'S HAZY TV, PDQ.
NEITHER: LOREM IPSUM DOLOR SIT AMET CONSECTETUR ADIPISCING ELIT.
PANGRAM: PACK MY BOX WITH FIVE DOZEN LIQUOR JUGS.
```

problem 6
Don’t Use No Double Negatives
5 points

Walking along the arena track, you almost don’t see the misdirecting sign for the event that seems to want to not be discovered near the water table. Then you realize the misinformation is part of the problem! There is an instruction sheet that explains:

A double negative is a sentence which uses two negative words, such as "no", "not", or "don't". Double negatives are often used to emphasize the negative, but in other cases they cancel each other. Some people consider their usage bad grammar. Even when used correctly, we won't disagree that double negatives can make a sentence confusing. More extreme usage may include three, or four, or more negatives in a single statement. Multiple negatives can be used for comic effect or to obscure meaning, but don't say we didn't never warn you not to do it.

Write a program to count the number of negatives in a sentence. Use this list of negative words provided below. The sample input will demonstrate that there may or may not be other forms of negatives we won't be failing to ignore.

DON’T CAN’T ISN’T HAVEN’T CANNOT WOULDN’T COULDN’T WON’T NO NOT NEVER NOBODY NOWHERE NEITHER AIN’T

Input

Each line of input is a single sentence that ends with a period. Sentences may have one or more of the following punctuation marks: , ; : ? ! "

The input ends with a single period.

```
THERE NEVER WAS NO MAN NOWHERE SO VIRTUOUS.
AXE YA EX WHAT TALKS IN TONGUES, SAY "NEVER MO NEITHER ME".
I HAVEN'T NEVER OWED NOTHING TO NO ONE.
BADGES? WE AIN'T GOT NO BADGES.
THIS IS A PERFECTLY POSITIVE SENTENCE.
I CAN'T GET NO, NO, NO, NO, HEY, HEY, HEY, NO SATISFACTION!
IT WOULDN'T BE INACCURATE TO ASSUME I COULDN'T SAY THAT I DON'T KNOW WHERE HE'S NOT.
.
```

Output

The program should print the number of negatives in each sentence, followed by a colon, space, and the sentence.

```
3: THERE NEVER WAS NO MAN NOWHERE SO VIRTUOUS.
2: AXE YA EX WHAT TALKS IN TONGUES, SAY "NEVER MO NEITHER ME".
3: I HAVEN'T NEVER OWED NOTHING TO NO ONE.
2: BADGES? WE AIN'T GOT NO BADGES.
0: THIS IS A PERFECTLY POSITIVE SENTENCE.
6: I CAN'T GET NO, NO, NO, NO, HEY, HEY, HEY, NO SATISFACTION!
4: IT WOULDN'T BE INACCURATE TO ASSUME I COULDN'T SAY THAT I DON'T KNOW WHERE HE'S NOT.
```

problem 7
Greek Acrophonic Numerals
6 points

You meander over to the next event and see a tapestry full of Greek numerals. The event coordinators explain:

The earliest alphabet-related system of numerals used with Greek letters was the set of acrophonic Attic numerals. These numerals operated much like Roman numerals (which derived from this scheme), with: I = 1, Π = 5, Δ = 10, H = 100, X = 1000, M = 10000. Also, 50, 500, 5000, and 50000 were represented by composites of Π and a tiny version of the applicable power of ten.

You must write a program to convert between Greek Acrophonic Numbers and our familiar decimal number system.

For this problem we'll use a numeral scheme that follows the pattern of this ancient Greek system. We'll represent the numerals like this: I = 1, P = 5, D = 10, H = 100, C = 1000, and M = 10000. For values above ten, groups of five are represented by a P followed by another letter. For example, 6 is written PI, 50 is PD, and 477 is HHHHPDDDPII.

Input

The first line of input will indicate the number of conversions the program must perform. Each line after will contain either a Greek acrophonic number or a decimal number.

```
9
8
50
475
CCPHHDDDDPII
CCCPHHHHPII
PMMCPDDDDDP
5678
PMMPCPHDDDPI
8642
```

Output

For each input value the program must convert from decimal to Greek acrophonic, or vice-versa.

```
PIII
PD
HHHHPDDDP
2647
3807
61095
PCPHHPDDDPIII
65536
PCCCCPHHDDDDII
```

problem 8
N-Ultimate
6 points

You saunter into the next pavilion to see a seemingly endless series of words and numbers being displayed on the walls. You read this explanation on the central table:

The last element in a series may be called the *ultimate* element. The *penultimate* element is next-to last. So, by extension, the *N-ultimate* element is the Nth element from the end.

Write a program to find the N-ultimate element in a series.

Input

Each line of input starts with an integer N, followed by a series of N or more words/numbers/strings, terminated with a $ symbol. The input ends with the number zero and a $ symbol.

```
4 PROXIMATE DISTANT EXTREME FARTHEST ULTIMATE $
6 999 0 426 123 1337 31415 1414 5 321 $
2 WHO WHAT WHEN WHERE WHY HOW $
3 RED GREEN BLUE YELLOW ORANGE PURPLE BLACK WHITE $
7 GARCIA WANG ZHANG LI SMITH MULLER GONZALEZ SMIRNOV NGUYEN HERNANDEZ $
0 $
```

Output

For each line of input the program must print the N-ultimate word.

```
DISTANT
123
WHY
PURPLE
LI
```

problem 9
Goldbach's Conjecture
6 points

You hike up the steps into the bleachers to discover the next event. For some reason, all the seats in this section are numbered with only prime numbers. In the neighboring section the seats are all even numbers. The managing event coordinator explains how these numbers relate:

Goldbach's conjecture says that every positive even number greater than 2 is the sum of two prime numbers. This conjecture has never been proven in the general case, but it has been confirmed for numbers much larger than most programming environments' native data types.

Write a program to print the two prime numbers that sum to a given even integer. There will often be more than one answer, so print the solution with the minimum difference between the two prime numbers.

Input

Each line of input will be a positive, even integer greater than two, except the last line, which will be zero. The maximum input value will be 1000.

```
28
62
992
16
0
```

Output

The program must print the two prime numbers and the sum in equation form in ascending numeric order. Print only the solution with the minimum difference between the two prime numbers.

```
11 + 17 = 28
31 + 31 = 62
421 + 571 = 992
5 + 11 = 16
```

problem 10
Decode This!
6 points

You dash to the closest event where the title appears to be in code:

DO TEE!ISCDH

The event coordinators explain how the title was created:

DECODE THIS! … becomes … DO TEE!ISCDH

There are 12 characters in both the original and encoded strings. The original first character is placed in the first encoded position. Each successive character is placed in an empty position based on the value of the last character:
Dxxxxxxxxxxx. D=4, so the next character, E, is placed in the 4th empty position after D:
DxxxExxxxxxx. E=5, so the next character, C, is placed in the 5th empty position after E:
DxxxExxxxCxx. C=3, so the next character, O, is placed in the 3rd empty position after C, wrapping:
DOxxExxxxCxx. O=15, so place D in the 15th empty position after O, wrapping again.
DOxxExxxxCDx. D=4.
DOxxEExxxCDx. E=5.
DO xEExxxCDx. The space = 1.
DO TEExxxCDx. T=20.
DO TEExxxCDH. H=8.
DO TEExIxCDH. I=9.
DO TEExISCDH. S=19 (but there's only one spot left for the "!" anyway.)
DO TEE!ISCDH

Values:
A-Z = 1-26.
a-z = 1-26.
Any other characters (punctuation, numbers, spaces) have a value of 1.

Input
Each line will hold a number N, then one space, then N characters to decode. N will be at most 80. The last line will have a single 0.

```
12 Do Tee!iscdh
25 Yotei! mcgaeos'rued a drn
0
```

Output
Each line's decoded string on its own line.

```
Decode This!
You're a decoding master!
```

problem 11
Chinese Remainder Theorem
7 points

You march into the next tent and quickly see many integers covering the walls, in an elegant calligraphy. A large portrait of the Chinese mathematician Sun Tzu decorates one wall. The one word “Remainders” is the title for a set of instructions:

In the third-century AD, Sun Tzu proposed a very powerful theorem, where numbers are represented as a set of simultaneous remainders. For example, if we use the base divisors 3, 5, and 7, we can uniquely identify any integer from 0 to 104 using only the remainders from those divisors. This is called "modulo arithmetic", where only the remainder is important to keep.

For example, when divided by [3, 5, 7], we represent a number as a set of remainders (x, y, z):

- the number 4 becomes (1, 4, 4).
- the number 20 becomes (2, 0, 6).
- the number 47 becomes (2, 2, 5).

The power of the theorem is made obvious when arithmetic is performed.

- For addition: 20 + 47 would be considered as (2, 0, 6)+(2, 2, 5) = (4, 2, 11), which becomes (1, 2, 4) when we divide by the original bases. And the number 67 = (1, 2, 4).
- Similarly for multiplication: 4 x 20 becomes (1, 4, 4) x (2, 0, 6) = (2, 0, 24), which reduces to (2, 0, 3) = 80.
- The operations can be performed on the smaller remainders and still represent the right answer for larger numbers.

The theorem works for any set of divisors [a, b, c] which are mutually prime (a, b, and c don't share common factors.) Your program will not need to do arithmetic, but will only need to identify the number represented by a set of remainders.

Input

Each line of input holds six integers "a b c x y z" separated by spaces. Each is less than 1000. The last line of input will be six -1s.

```
3 5 7 2 0 6
7 15 16 3 2 1
23 49 96 3 30 77
127 541 59 17 120 15
21 23 40 0 0 0
-1 -1 -1 -1 -1 -1
```

Output

For each line, you must print the lowest positive integer N that meets the requirements:

- remainder(N/a) = x
- remainder(N/b) = y
- remainder(N/c) = z

N will have at most 6 digits.

```
20
17
98765
999888
19320
```

problem 12
Boustrophedon
7 points

In a portion of the arena field, a fence has been constructed to hold a large ox. The sign over the pen has a title reading:

```
##### #   #  ###   ###       # ##### #   #
    # ## ## #   # #   #      #     # #   #
##### # # # #   #     #      # ##### # # #
    # #   # #   # #   #      #     # # # #
##### #   #  ###   ###  ##### #####  # #
```

You cautiously creep nearer, careful not to disturb the animal, and read the stone tablet near the pen:

The Phonecians invented the alphabet and wrote right-to-left. The Greeks learned to write from the Phoenicians and at first they also wrote right-to-left. Over time the Greeks developed a writing style called Boustrophedon in which lines of text alternated right-to-left and left-to-right. Additionally, they wrote their letters in alternating directions as well, so that the letters "faced" the direction of the writing. There was no punctuation and no spaces between words. Boustrophedon means "ox-turning", and the style of writing is so named because it mimics a farmer plowing a field with an ox, back and forth in rows. Write a program to decode English text written in boustrophedon.

Input

The first line of input will indicate the number of rows of text to follow. After that, the input consists of a series of lines of English text written in boustrophedon. Each letter is represented by a five-by-five grid of spaces and hash "#" symbols. Each letter grid is separated from the next grid by a single space. A blank line is provided before each row. The program must determine if the direction of the first line is right-to-left or left-to-right. It could go either way. All following lines alternate in direction. There will be no more than twelve letters per row.

2

```
##### #   # ##### ##### ##### ####   ###  ##### ####    #   #   # #   #
  #   #   # #     #       #   #   # #       #   #   #  # #   # #  #   #
  #   ##### ##### #####   #   ####   ###    #   #   #  ###     #   # # #
  #   #   # #     #       #   #  #      #   #   #   # #   #    #   # # #
  #   #   # ##### #     ##### #   #  ###    #   ####  #   #    #    # #

      #####  ###    #   #####  #### #####     # #####   #    ###  #####
        #       #  # #      # #   #     #     #   #    # #      #      #
        #    ###   ###  ##### #   # #####     #   #    ###   ###  #####
        #   #     #   #     # #   #     #     #   #   #   # #         #
        #    ###  #   # #####  #### ##### ##### ##### #   #  ###  #####
```

A file named **letters5x5.txt** provides the left-to-right forms of the letters. Your program may read the file, or you may copy the data into your program. The file is a sequence of twenty-six five-by-five grids. Spaces are represented by periods and lines are represented by the English letter represented in the grid. Two examples are shown below.

```
RRRR.
R...R
RRRR.
R..R.
R...R
.SSS.
S....
.SSS.
....S
.SSS.
```

Output

The program must transcribe each row of input into English letters and print the letters as a single line of output. All output must be printed left-to-right (not boustrophedon).

```
THEFIRSTDAYWESAILEDEAST
```

problem 13
Five Letter Words
8 points

Your team boards a small cart which rolls into a chamber under the arena's bleachers. Inside the room is a tall door with an electronic lock. On its display is a series of random letters and a set of instructions:

> *Take these letters and combine them into all possible 5-letter "words". Alphabetize them and remove all duplicates. Number your final list starting with 1. Then type in the words corresponding to the numbers shown on the screen. If all of your entries are correct, the door will open for you to receive your reward.*

Write a program to find the correct 5-letter entries for the letters and list-locations provided.

Input

The input starts with an integer N (from 1 to 10). The next line holds N capital letters separated by spaces. They may not be unique, nor in alphabetical order. The following lines each have one integer K. The input ends with a zero (0).

```
8
B R E D E A E A
1
3
316
321
0
```

Output

Arrange the letters into all possible 5-letter combinations ("words"). You may use each letter only as often as it is appears in the input. Arrange these words in alphabetical order, eliminating duplicate words. For each integer K, print the value of K, followed by a colon and the K'th word in the alphabetical list (the first word is number 1.) If K is larger than the number of list entries, print the list length and last entry in the list.

```
1: AABDE
3: AABED
316: BREAD
321: BREED
```

Example Input 2

```
10
B A A A A B B B B A
1
16
40
0
```

Example Output 2

```
1: AAAAA
16: ABBBB
32: BBBBB
```

It may be helpful during development to print your list to make sure you've eliminated all duplicates and your list is ordered correctly.

Problem 14
Buying Beans
8 points

You walk over to the next event where you see several vines of beans have been planted. The event coordinators (oddly dressed as arena gardeners) offer you a curious trade:

You may pick any number of beans you like and place them on one side of a balance scale. The gardeners will then place weights on the other side of the scale. If they CANNOT make the scale balance, they will award you one arena-coin for each of your beans. But you will only be awarded your points if you earn the maximum number of coins.

The gardeners use a different set of base weights for each weighing, so you will always have to work out your best deal. Each bean weighs one gram, and the weights are also in units of grams.

For example, if the gardeners have a supply of weights with values of 5 and 7 grams, they cannot balance 1, 2, 3, 4, 6, 8, 9, 11, 13, 16. 18 or 19 beans. And the largest number they cannot balance is 23 beans.

If they next use weights of 6, 7, and 11 grams, the largest number they cannot balance is 16 beans.

Write a program to determine the largest number of beans which cannot be balanced for a given set of weights.

Input

The first number on each line will be an integer N (2, 3, or 4), representing the number of different weights the gardener uses. The rest of the line holds N different integers, the value in grams of each weight. The maximum base weight is 100 grams. The last line holds a single 0.

Example 1:
```
2 7 5
3 6 7 11
0
```

Example 2:
```
4 11 12 13 14
2 99 2
2 99 100
0
```

Output

Print the maximum number of beans which cannot be balanced using any combination of multiples of the N weights. [In cases when all the weights share a common factor F, the maximum is infinite because the weights cannot combine to any value equal to 1 plus a multiple of F. You may assume the input will never include these cases.]

Example 1:
```
23
16
```

Example 2:
```
43
97
9701
```

Note: To help limit any programming loops you might use, the maximum number of beans will never be greater than 10000. It will usually be much smaller.

problem 15
Fruit Math
10 points

You glance into the second refreshment tent and are pleased to find fruit on the tables, but before you can grab any, you're told you must first determine the value of each fruit.

The fruit is arranged in rows and columns, where each kind of fruit represents a unique digit between one and nine. Given sums for each row and column, determine the value for each type of fruit.

The nine fruits that can be used in each puzzle are:
Apple, Banana, Cherry, Durian, Fig, Grape, Lychee, Orange, and Pear.

Input

The first line of input indicates the size of the square fruit grid, up to a maximum of nine. Each row after that lists the fruits for a single row, followed by the sum for the row. The last line contains the sums for the columns. Please observe that extra spaces may be inserted between the items to line up the columns for visual reference. Any line might begin with a space.

```
4
  D  L  D  L 14
  F  P  F  D 14
  F  L  L  P 22
  D  P  L  L 20
  8 26 16 20
```

Output

The program must print the value and name of the fruits in the puzzle in numeric order. There is only one solution.

```
1 Durian
3 Fig
6 Lychee
7 Pear
```

problem 16
Unfinished Travel
12 points

During the competition in the HP CodeWars Arena today, you are travelling [virtually] by many means (jogging, walking, etc.) For this event, you must find those modes of transport. They are hidden in the grid of letters below, but some of the words are missing a letter.

Your program must replace all of the digits in the input with appropriate letters so that all of the provided words can be found in the grid in a word-search fashion (horizontally or vertically, in any direction, but NOT diagonally.) Letters may be used in more than one word, but missing letters are used in exactly one word, and no word is missing more than one letter. Careful! Each digit could represent different letters to complete different words, but only one set of replacements will result in all words being found. Each word will be found exactly once in the final grid.

Input

The first line contains two integers X Y, representing the number of rows and columns in the data. Maximum of 10 rows and 20 columns. The next X lines include the grid of letters, Y characters at a time. It consists of only capital letters (A-Z) and numbers (1-9) marking the location of missing letters. After the grid is a line with a single integer N (less than 20), then N lines with the words to be found in the search grid.

```
5 13
SLH8UROW3LIDE
92OGAMEANDERL
UJPCRA5LUGCO4
R7TNUASK6PALM
CGXY1LIMBERLA
14
JOG
RUN
HOP
WALK
RUSH
RACE
CLIMB
AMBLE
SLIDE
SAUNTER
MEANDER
CRAWL
SKIP
ROLL
```

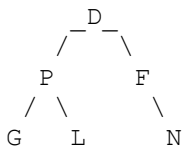
Output

Print in numerical order one line for each digit, its replacement letter, and the word it is in, each separated by one space.

```
1 C CLIMB
2 J JOG
3 S SLIDE
4 B AMBLE
5 W CRAWL
6 I SKIP
7 E SAUNTER
8 S RUSH
9 N RUN
```

problem 17
Depth First
12 points

You weave your way through several trees at the edge of the arena, finding various letters hanging in a pattern representing a binary tree. Every node in a binary tree has some kind of value and it can also have references to two child nodes, called left and right. Either or both child references may be empty. The top node in a tree (or subtree) is called the parent node or head node.



In this diagram, D is a parent of two children: P is the left child and F is the right child. Node F is the parent of one child, N, on its right, while F's left child reference is empty. Node P is the parent of G and L. Nodes G, L, and N each have two empty child references.
Node processing can be done in a variety of ways. For this program, the goal is to process the left child first, then the right child, and then the parent. This type of traversal is called post-order. Using a post-order traversal, the nodes in the example tree would be processed in the following sequence:

G L P N F D

Input

The first line of input indicates the number of subtrees in the input. Each line after is a simple subtree. Each subtree line has three letters, separated by spaces, in this order: parent, left, then right. Empty child references are represented by a period. The subtrees are not given in any particular order. You'll have to determine the head of the tree using cleverness and ingenuity, which are not included in the input.

```
7
O . L
C O W
S C G
G I N
T A R
I H T
L F .
```

Output

The program must print the post-order traversal sequence of the nodes.

```
F L O W C H A R T I N G S
```

problem 18
Sapphire Search
13 points

You hop over a low barricade to discover an archeological dig at the side of the arena. Among the artifacts on display, there is a curious note:

> *I buried my sapphire then started walking. I always walked in a straight line following a compass direction (N, S, E, W). When I stopped, I made a 90 degree turn and continued walking. I might have crossed my path, but I don't remember. Below are the number of meters I travelled in each direction. I'm now lost and must abandon this record while I search for a way out. I'm placing this note under a rock at my final location. Perhaps some lucky adventurer will decode my note and retrace my steps to earn the treasure.*

Unfortunately, there is no record of where in the ruins the note was found. Instead, you must write a program to find the treasure.

Input

The first line contains two integers X Y, representing the number of rows and columns in the ruins. Maximum of 20 rows and 50 columns. The next X lines show a grid map of the space. A period "." is an empty square. A hash "#" is a large boulder, marking a square that cannot be entered. The next line has an integer N, the count of the straight paths walked. Maximum of 20 paths. The last line contains N integers separated by spaces, showing the successive path-lengths..

```
5 10
##########
#........#
#.#...##.#
#...#....#
##########
8
2 4 2 2 2 5 2 1
```

Output

Your program must print the same map, with the location of both the Sapphire (S) and the final location of the message (F) marked. Also, label every turning point with successive lowercase letters (if the same point is used more than once, print the letter for the later turn.) There is only one route which follows the path-lengths in the list.

```
##########
#b.e.a..f#
#.#...##.#
#c.d#S.Fg#
##########
```

problem 19
Serpentine Sentence
15 points

You climb the many stairs to the arena's broadcast booth where the code-maker is again selling her creations. She continues to say her codes can't be beat (only to be proven wrong.) This year, she has made her creations even more complex:

> Serpentine Sentence: *Discover codes hiding around riddles. Leading a serpentine trail, I negate programmability. Use today!*

Hmph! She says she can't be beaten by a program. You need to prove her wrong! We know that she tends to repeat herself, and she uses long sentences. So in each code she makes, we'll find exactly 2 copies of the data, and it will be the longest repeated data we find. The sentence is written into a grid of characters in a serpentine fashion, moving up, down, left or right (not diagonally), and it never returns to touch its own path (that is, the sentence does not turn so that a later character is directly adjacent to an earlier one.) Somewhere else in the grid, the sentence is repeated in exactly the same way as the first, moving in exactly the same directions. The two copies may overlap each other. There may be other repeating sets of characters in the grid, but the longest paired repeat is the sentence we seek.

Your program must search the grid for the longest series of characters that can be found identically in two places. The longest possible length of the sentence is 80 characters.

Input

The first line contains two integers X Y, representing the number of rows and columns in the data. Maximum of 20 rows and 50 columns. The next X lines include the coded message, Y characters at a time. It consists of letters and numbers (A-Z, a-z, 0-9), punctuation and spaces.

```
12 26
IAI my MatC76 Trombones le
 FkeCIDIch bin ein BerliId
wGiHIJtiw ImpressiverlInI
aIl IMhNIOI!edarap gib eht
nZ!aQR Invent! my Mat wriS
tITzUIaWX ofZkeYIZIAhIaCI?
 piz.E IJeA Pi.KLtiw MnX!a
I win slicIOPl IQhRSTUtVWz
76 Trombones leXI YZAB piz
Ich bin ein BIdCIaEI ofXII
ImpressiveHIer JK LIeB Pi.
!edarap gib ehtNIslicI win
```

Output

Print the discovered sentence. Since we can't be sure in which order the characters appear, print it both forward and backward.

```
I like my Math with a slice of Pi.
.iP fo ecils a htiw htaM ym ekil I
```

This sentence is hidden in the grid in this way:

```
  my Mat
ke     h
i   tiw
l I h
             my Mat
    a   of ke     h
       e  Pi.  tiw
    slic   l I h

               a   of
                  e  Pi.
               slic
```

problem 20
Product Distribution
15 points

You skip into the center of the arena, where several crates have been placed in rows. The event coordinators explain that the grid of crates will be obscured when they raise tarps around them. The crates will then be moved and your task will be to determine the new configuration from a few clues. There are two rules the configuration will always follow:

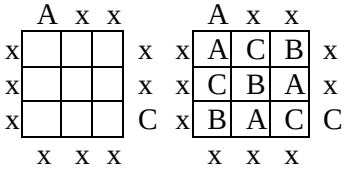
- The layout is a square grid, 3, 4, or 5 meters wide. Each crate fits in one of the 1-meter squares.
- There are three types of crates, labeled A, B and C, and exactly one of each crate will be found in each row and column.

The event coordinators raise the tarps and modify the configuration. For clues, they have positioned cameras at the end of every row and column. Each camera shows the first crate it sees from its position along the row or column. Some of the cameras have been turned off, but there is enough information to determine the arrangement.

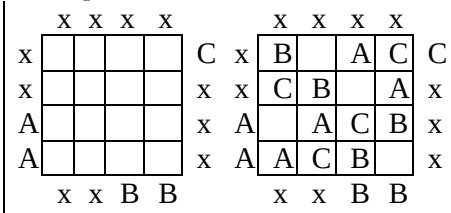
You must to write a program to use the views from the cameras and print the distribution of the crates.

In the following examples, "x" represents a camera that is turned off. A/B/C each represent a camera that is on and the type of crate it sees first in its row or column.

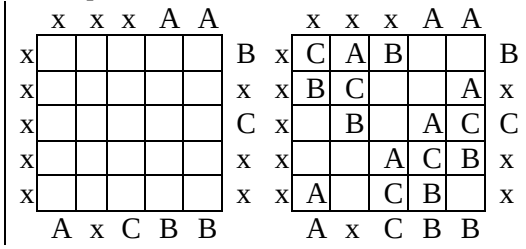
Example 1: 3x3.



Example 2: 4x4.



Example 3: 5x5.



Input

The input is first an integer N representing the size of the square grid. The next line shows the first crate seen in each column from the top of the grid separated by spaces (X means the camera is off.) The next N lines show the first crate seen on each row from the left and right of the grid. The last line shows the first crate seen in each column from the bottom of the grid.

Example 1
```
3
A X X
X X
X X
X C
X X X
```

Example 2
```
4
X X X X
X C
X X
A X
A X
X X B B
```

Example 3
```
5
X X X A A
X B
X X
X C
X X
X X
A X C B B
```

Output

Print the NxN grid, with space between each square in a row. Use a period for empty squares.

Example 1
```
A C B
C B A
B A C
```

Example 2
```
B . A C
C B . A
. A C B
A C B .
```

Example 3
```
C A B . .
B C . . A
. B . A C
. . A C B
A . C B .
```

problem 21
Different Floors
20 points

Looking for a little rest, you hike across the arena parking lot to the adjoining apartment building where competitors are housed, only to discover another challenge. The sign next to the elevator has lost its names for each floor. Instead, a set of clues have been provided:

Morales, Chen, Tanaka, Patel, and Smith live on different floors of a building with five floors. Morales does not live on the 5th floor. Chen does not live on the 1st floor. Tanaka does not live on either the 1st or the 5th floor. Patel lives on a higher floor than does Chen. Smith does not live on a floor adjacent to Tanaka's. Tanaka does not live on a floor adjacent to Chen's. Where does everyone live?

Write a program to solve logic problems like this one. The program will read exactly six statements, where each statement is one of these types:

- <name> NOT ON FLOOR <x>
- <name> NOT ON FLOORS <x> OR <y>
- <name> ON HIGHER FLOOR THAN <other name>
- <name> ON ADJACENT FLOOR TO <other name >
- <name> NOT ON ADJACENT FLOOR TO <other name >

Hint: It's a good idea to see if your program finds multiple solutions. You could have a nasty bug that sometimes incorrectly finds more than one solution. If the program finds the right one first with the sample data but an incorrect solution first with the judge data, you may have a lot of trouble finding the cause.

Input

The input will consist of six sentences, each on a separate line. There is no guarantee that any particular statement type will appear. A statement type may appear several times.

Example 1

```
MORALES NOT ON FLOOR 5
CHEN NOT ON FLOOR 1
TANAKA NOT ON FLOORS 1 OR 5
PATEL ON HIGHER FLOOR THAN CHEN
SMITH NOT ON ADJACENT FLOOR TO TANAKA
TANAKA NOT ON ADJACENT FLOOR TO CHEN
```

Example 2

```
BLAKE NOT ON FLOORS 5 OR 4
GONZALEZ ON ADJACENT FLOOR TO ITO
HUANG ON HIGHER FLOOR THAN PHAM
BLAKE ON HIGHER FLOOR THAN HUANG
GONZALEZ NOT ON FLOOR 1
ITO NOT ON ADJACENT FLOOR TO BLAKE
```

Output

The program must print the five floor numbers with the names of the occupants, from top to bottom.

Example 1

```
5 PATEL
4 TANAKA
3 MORALES
2 CHEN
1 SMITH
```

Example 2

```
5 ITO
4 GONZALEZ
3 BLAKE
2 HUANG
1 PHAM
```