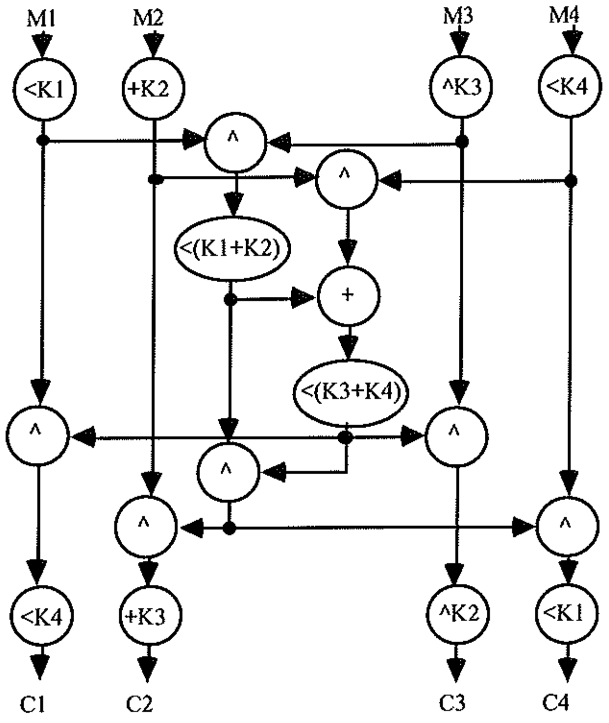
PROBLEM A Wily Hacker's Problem

You have been assigned the job of checking the country's security by trying to break the DES-like encryption scheme which is used for secret messages. There is a bonus of $100,000 if you do manage to crack it. Even though you have some keys and some samples of encrypted messages, the job looks impossible because you don't have the algorithm. Then you have a stroke of luck; working late one night, you see on the chief's desk a diagram of the encryption scheme used, and after a moment's thought you see that there is a fatal flaw, and the scheme is easy to crack. Here is the diagram:



You know that the standard used is to encrypt a 64-bit message M (divided into four 16-bit parts M1, M2, M3, M4 in order from left to right) into a 64-bit cryptogram C (divided into four 16-bit parts C1, C2, C3, C4) under the control of a 64-bit key K (divided into four 16-bit parts K1, K2, K3, K4), and that the encryption function uses 3 types of operations:
• exclusive-or, marked with the symbol ^;
• addition modulo 2^16 (ie ignoring overflow), marked with the symbol +;
• circular left shift marked with the symbol <K, where K is the number of bits moved (eg the 16-bit word 1010010001000010 under the operation <5 becomes 1000100001010100).
All these functions operate on 16-bit operands.

The plaintext and encryptions which you know are given in the examples below.

The input is from the file PROBLEMA.DAT and consists of several pairs of cryptograms and keys. Each line consists of a cryptogram, composed of 16 hexadecimal digits (64 bits) and, after one or more blank characters, the 16 hexadecimal digits (64 bits) of the encryption key used to produce that cryptogram. The message is terminated by a line consisting of a single # character.

For each line of the input file, your program must write the ASCII characters (note that each ASCII character represents two hexadecimal characters) of the messages (plaintext), representing the result of decrypting the cryptogram on the corresponding line of the input file. You may assume the decryption will always result in a set of printable ASCII characters.
EXAMPLE
Sample Input
```
6742a447d483868e 1111222233334444
4ced98bf8eb91895  feedf1d0facef00d
#
```
Output for this Input
```
STUDENTS
TEACHERS
```

PROBLEM B Mastermind

Suppose we have a board with 5 holes in which to place coloured pegs chosen from 8 colours. Some of the pegs may be the same colour.

One player (secretly) chooses the colours of the pegs to be placed in the 5 holes. The other player has to guess the colours of the pegs. For each guess, the second player is told the number of pegs whose colour and hole is guessed correctly, and the number of pegs whose colour is guessed correctly, but the hole is wrong. For example if the board has colours 2 6 5 6 4 and the guess is 1 1 2 3 4, then the reply will be that there is 1 peg in the right place, and 1 of the right colour.

Intelligent programs can be written to determine the solution in 6 queries or less. Your job is **not** writing such a program (I'm sure you could do it we'll leave it for another year) but to provide a program for checking if solution programs really work. You will be given a set of guesses and the responses, and you must find if some layout of the colours is compatible with the guesses and responses, and whether this solution, if it exists, is unique.

Input will be from a file PROBLEMB.DAT and will consist of sets of data. Each line of data contains 7 integers separated by one or more spaces; the first 5 give a "guess" of the 8 colours for the 5 pegs and are integers from 1 to 8 inclusive, and the last two give the response; the first number counts the number of completely correct guesses and the next number counts the pegs whose colours have been guessed correctly, but not their positions - these last two numbers will be from 0 to 5 inclusive. Each data set is terminated by a line consisting of seven 0's; there will be no more than 10 lines in a dataset. The file will be terminated by an empty data set (ie a second line of 7 0's).

Output will be one line for each data set, which will either be 5 digits separated by single blanks, giving the 5 colours of the problem solved by the data set, or the words "No solution" or "Non-unique solution", if no solution, or a non-unique solution, can be found.

EXAMPLE

Sample Input

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

Output for this Input

```
2 6 5 6 4
Non-unique solution
No solution
```

PROBLEM C Coins

In a certain country, it has been decided to do away with banknotes as much as possible. The basic coin is a $1 coin. The coins increase in value, and each coin has 10% larger radius than the next largest coin (they all have the same thickness and are made of the same material). It is desired to make sure that as many values as possible can be made by presenting coins, up to a weight of N single dollar coins. For example, if N = 4.5 and there are two coin denominations, $1 and $4, then any amount up to $10 can be made from coins weighing, in total, less than 4.5 dollar coins, but $11 can't be so made, so the smallest banknote must be $11.

The maximum weight N has not been decided (but it is definitely no more than 8), nor has the number of different coin denominations (but it is definitely no more than 4). Your job is to write a program so that various values can be worked out. Input will be from the file PROBLEMC.DAT and will contain lines describing several different scenarios. Each line contains two numbers, separated by one or more spaces. The first number gives N, the maximum weight of coins allowed (in terms of the $1 coin), which could have a fractional part (no more than one decimal place). The second number is an integer giving the number of different coins. The list of scenarios is terminated by a line containing two zeroes.

For each input line you must output a line describing the solution which leads to the largest possible value for the smallest banknote. If more than one solution produces the same value for the smallest banknote, you must output the solution which leads to the smallest value for the coin of greatest value; if there is still a non-unique solution, the one which gives the smallest value for the next largest coin, etc. Your solution must list the value of the smallest banknote, then the values of the coins in decreasing order (ending with 1 for the $1 coin).

EXAMPLE

Sample Input

```
5 2
4.5 3
7.5 4
0 0
```

Output for this Input

```
11 3 1
19 8 5 1
81 25 18 5 1
```

PROBLEM D Contour Painting

A shape is represented by drawing its outline contour in a two dimensional grid, as illustrated in the figure below. The points of the contour are specified by the same character which can be any printable character except a space. In the figure below this character is X. All the other points inside and outside the shape are represented by spaces. The contour is connected, ie any two points on the contour can be reached from one another by travelling vertically, horizontally and diagonally. There will always be some empty spaces inside the contour. The aim of this problem is to "paint" either the outside of the contour or its inside.

```
 XXXXXXXXXX
XXXX       XX
X            X
X      X     XXXXXXX
XXXXXXXX            XX
X      X     XXXXXXX
X            X
XXXX       XX
 XXXXXXXXXX
```

The contour can always be broken into regions which are straight, a concave corner, or a convex corner, and in the figures below these are "painted" with the character *. Note that the "paint" is added on one side of the contour in such a way that every horizontal or vertical neighbour of each contour point is either another contour point or a *, and the least possible number of *'s is used.

```
                          ****
***          X***         XXXX*
XXX          XXXX            X*
```

straight concave corner convex corner

A contour can be painted either from inside or from outside. The painting side is specified by the presence of the character, which is different from space and from the character used for the contour, inside or outside the contour as shown in the figures below. This character is used for the painting, and it is removed from the grid once the painting is done (or becomes part of the painting).

```
 XXXXXXXXXX
XXXX       XX
X *          X
X      X     XXXXXXX
XXXXXXXX            XX
X      X     XXXXXXX
X            X
XXXX       XX
 XXXXXXXXXX
```

before painting

```
 XXXXXXXXXX
XXXX*******XX
X***   *   **X
X******X*   *XXXXXXX
XXXXXXXX*    *******XX
X******X*   *XXXXXXX
X***   *   **X
XXXX*******XX
 XXXXXXXXXX
```

after painting

interior painting

```
 XXXXXXXXXX     *
XXXX       XX
X            X
X      X     XXXXXXX
XXXXXXXX            XX
X      X     XXXXXXX
X            X
XXXX       XX
 XXXXXXXXXX
```

before painting

```
  **********
 *XXXXXXXXXX**
*XXXX       XX*
*X            X******
*X      X     XXXXXXX**
*XXXXXXXX            XX*
*X      X     XXXXXXX**
*X            X******
*XXXX       XX*
 *XXXXXXXXXX**
  **********
```

after painting

exterior painting

Your program must read contours from a file named PROBLEMD.DAT. Each contour will be constructed from a single symbol and the contours will be separated from each other by two lines which are either blank (no characters or with just space characters) or have a painting symbol for the upper/lower contour. The leftmost character of each line will never be a contour symbol. Each contour will contain a single painting symbol. There are at most 30 lines (including the starting and ending lines with no contour symbol) and at most 80 characters in a line for each contour. The file will be terminated by an empty contour - this will be two adjacent blank lines after a line which is blank or has a single painting symbol..

Your program must paint each contour with the painting symbol, interior or exterior according to the position of the painting symbol, and output the painted grid. Your output must be exactly the same as the input except that some of the space characters have been replaced by the appropriate painting symbol and the painting symbols have been removed (if necessary).

EXAMPLE
Sample Input

```
 XXXXXXX
 X  *  X
 XXXXXXX


   Q X
  Q Q
 QQQQQ
```

Output for this Input

```
 XXXXXXX
 X*****X
 XXXXXXX

   X
  XQX
 XQ QX
XQQQQQX
 XXXXX
```

PROBLEM E Trains

Two towns T1 and T2 are connected by a double railroad. The distance between T1 and T2 is d meters. From T1 to T2 trains leave every t1 seconds. From T2 to T1 trains leave every t2 seconds. The trains from T1 to T2 have a speed of v1 m/s. The trains from T2 to T1 have a speed of v2 m/s.

Your task is to write a program that computes the number of train "rendezvous" on the railroad which links T1 and T2, and which occur during the time interval [0, tf] seconds. A train "rendezvous" occurs if two trains pass each other while heading in opposite directions, or one arrives at a station at the exact instant that another train leaves that station.

We consider that:
a) at time 0 two trains are leaving (from T1 to T2, and from T2 to T1);
b) the input and output data are integers.

Input will be from a file PROBLEME.DAT and consists of lines, each of which contains a single data set in the format: d v1 v2 t1 t2 tf. There could be any number of blanks between the numbers on each line The file will be terminated by a line containing a single number 0. You may assume tf, t1 and t2 are chosen so there are no more than 100,000 train departures (note that there are 86,400 seconds in a day). You may also assume "natural" limits on d, v1 and v2; d < circumference of Earth, v1 and v2 < speed of sound.

Output will be one left-justified number for each line of input, giving the number of train "rendezvous" for that data set.

EXAMPLE
Sample Input
```
10 5 5 1 1 2
11 3 2 4 3 13
10000000  157 83 43 41 2000000
0
```

Output for this Input
```
6
9
201493194
```

PROBLEM F   Numbers

You have been contacted by a numerologist who needs your help to solve a basic problem of numerology. Everyone knows that the true meaning of a word can be found by adding up the value of each of its letters. This method will unlock the secrets of the Universe, if only people could work out what language to use, and how to work out the value of each letter. The numerologist has solved the second problem, and needs your help to solve the first.

His idea is to find the value of the letters by working on the words for numbers. The "true meaning" of the name of a number must be the value of the number. For example, in English, the true meaning of the word "SIX" must be the number 6, ie

$S + I + X = 6,$

where S, I and X are the values of the letters "S", "I" and "X". By finding values for letters so that we have:

$O + N + E = 1, T + W + O = 2, ..., N + I + N + E + T + Y + N + I + N + E = 99,$

the secrets of the Universe will be unlocked (he doesn't think we need to go past 99)! Unfortunately, after many years of work, he has been unable to find letter values (with every letter having a different value) which will work for numbers 1 to 13, and he suspects it is impossible to find any such values. What he has reluctantly concluded from this is that English is not the correct language. He has just heard of computers, and wants you to write a program which will find letter values which satisfy a set of number words written in some language. If he can find a language in which there are letter values which work for every number from 1 to 99, he will be famous (and live for ever - he is already rich from his consultancy work for governments).

Input will be from the file PROBLEME.DAT and will consist of several number - word sets for different languages. Each set will start with a line containing a number, giving N, the number of numbers for this attempt, followed by one or more spaces then the name of the language for this set. After this there will be N lines, each line having a number on it (less than 100) followed (after 1 or more blanks) by the name of that number in the language for this set (less than 60 characters - only the letters A to Z will be used). The file will be terminated by a line with an N which is 0.

For each number - word set, you must output the name of the language on one line, then either "Impossible" on the next line, or lines giving a set of values for the letters which will make the sum of the letters for each number word equal the number it represents. The letters used in the number words must be listed (one per line, in alphabetic order), then a space, then the value left-justified, as an integer if it is integral, otherwise rounded to two decimal places. Any solution will do, if more than one is possible. A solution is impossible if there is no set of values for the letters, such that each letter has a different value, which gives the number words the correct values.

EXAMPLE over the page

Sample Input

```
3 Test1
1 AB
2 BC
3 BACB
3 Test2
1 AB
2 BC
4 BACB
3 Test3
1 AB
2 BC
4 BACC
3 Test4
1 AB
2 BC
6 BACC
12 English1-12
1 ONE
2 TWO
3 THREE
4 FOUR
5 FIVE
6 SIX
7 SEVEN
8 EIGHT
9 NINE
10 TEN
11 ELEVEN
12 TWELVE
0
```

Output for this Input

```
Test1
A 1
B 0
C 2
Test2
Impossible
Test3
Impossible
Test4
A 1.50
B -0.50
C 2.50
English1-12
E 3
F 9
G 6
H 1
I -4
L 0
N 5
O -7
R -6
S -1
T 2
U 8
V -3
W 7
X 11
```

PROBLEM G Phone Words

Inspired by recent advertising, you have decided to write a program to interpret phone numbers as words so they can be remembered easily (for example "NUMBERS" for 6862377). Even better, you have realised that some money could be made this way - for a small fee, you will give people a "word" version (a mnemonic) of any phone number they give you.

To prevent competition and further increase your revenue, you have had the truly brilliant idea of making customised number-letter conversions. As a special bonus, you will give your customers a special mask, made just for them, printed with the letters you have chosen to associate with each digit. They can lay this mask on top of their push-button phone (there are holes for the buttons to come through) so that the letters for each digit are clear. Then, for each phone number they commonly dial, they can buy a mnemonic based on your letters. Since the masks are different for each customer, they will all have a different set of mnemonics for the same phone numbers (for example, 3257226 could be RAWMEAT for one customer and POISOND for another) and they won't be able to cheat by using mnemonics they haven't paid for (a mnemonic will always be written in upper-case letters with no spaces between the separate words).

For the customer with the RAWMEAT mnemonic, the mask could be:
0 = B,K,L,P 1 = F,G,Q 2 = A,E 3 = D,R,Z 4 = H,S,X
5 = J,W 6 = I,N,T 7 = C,M,V 8 = U,Y 9 = O
and for the customer with the POISOND mnemonic, the mask could be:
0 = K,W 1 = C,L,U 2 = N,O,T 3 = A,E,P,X 4 = F,J,Q,Z
5 = B,I,R 6 = D,G,M 7 = H,S 8 = (no letters) 9 = V,Y

Input will be from the file PROBLEMG.DAT and will be in two parts. The first part will be a dictionary of the valid words for this problem - there could be up to 1000 words in this dictionary; all words will be written in 7 or fewer upper-case characters, one word per line. The words **will be in alphabetic order**. This part of the input will be terminated by a line consisting of a single #. After this will be sets of Phone Mask-phone number combinations. Each combination will start with 10 lines, each line containing from 0 to 5 upper-case letters separated by 1 or more blanks, which represents the phone mask. The first line gives the letters (if any) corresponding to 0, the next the letters corresponding to 1, etc. Every letter will appear once and only once in these 10 lines; the letters will be in alphabetic order on each line. After this Mask description will be a set of 7-digit phone numbers, one per line. This list of phone numbers will be terminated by a line consisting of the number 0. Then there will be another Phone Mask-phone number set, and so on. The Phone Mask-phone number sets will be terminated by a line consisting of a single # (instead of a line giving letters for 0).

For each phone number in a Phone Mask-phone number set your program either give the word "Impossible" if no mnemonic can be designed for this number, or the mnemonics which can be constructed for the number from the words in the dictionary, listed in alphabetic order with a single space between each mnemonic. There must be no duplicates in the list. You may assume the words in the dictionary resemble a random selection of normal English words and abbreviations (the only single letters in the dictionary will be A, D, I and S). There will never be more than 10 mnemonics for a single number.

EXAMPLE over the page

Sample Input

```
A
ABLE
D
DON
MEAT
ON
POISON
RAW
RED
#
C K L P
F G Q
A E
J R Z
H S X
D W
I N T
B M V
U Y
O
3257226
0
K W
C U
N E L O T
A P X
F J Q Z
I R
B D G M
H S

V Y
3257226
3257228
3622530
0
#
```

Output for this Input

```
RAWMEAT REDMEAT
POISOND
Impossible
ABLERAW ADONRAW
```

PROBLEM H Hands together

A jeweller friend of yours has designed a truly beautiful clock with hands shaped like Mickey Mouse (for the minute hand) and Minney Mouse (for the hour hand). When the two hands cross each other (as they do once each hour, roughly), Mickey and Minney appear to kiss (isn't that sweet!). The mechanism is cleverly designed so that this action does not slow the clock down, but there is a certain amount of wear on the hands and the lifetime of the clock may be limited. She wants you to work out the number of times the hands will cross over a given time period (always less than a century).

Remember that there are 24 hours in a day and this is a normal 12-hour clock. There are 365 days in a year except for leap years (all years divisible by 4), when there are 366. The number of days in each month in a normal year is 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31; in a leap year, the extra day is 29th February. Daylight Saving is in operation in this state (sorry, Brisbane) and you may assume that it always starts on the 4th Sunday in October (10th month - 2 am on this day becomes 3 am; the hands will cross once as the clock is turned forward, assume they are turned forward exactly at 2 am) and comes off on the 2nd Sunday in April (4th month - 2 am on this day becomes 1 am; again the hands cross as they are wound back and assume this happens exactly at 2 am). Tomorrow is Sunday 15th October 1996 and Sundays occur every seven days exactly.

Input will be from a file PROBLEMH.DAT and will consist of lines each containing a starting time and date and an ending time and date. The starting date will always be in the future from today and the ending date will always be before 2100. The format of each time and date will be:
HH:MM DD/MM/YYYY where HH = hour (00 to 23; 0 = midnight, 1 = 1 am, ..., 12 = noon, 13 = 1 pm, .., 23 = 11 pm), MM = minute (00 to 59), DD = day of month (01 to 31), MM = month number (·01 to 12), YYYY = year (1996 to 2099). A midnight (00:00) time is assumed to be the start of a day - ie the minute after 23:59 31/12/1999 is 00:00 01/01/2000. Every number will be two digits (leading zeros if necessary), except for the 4-digit year. There could be any number of blanks on either side of the numbers and separating symbols (: and /). The two times and dates on each line will be separated by any number of blanks. An"impossible" time (between 2am and 3am on a day Daylight Saving starts) will never be given; if an ambiguous time (between 1am and 2am on a day Daylight Saving stops) is given, treat is as the first occurrence of this time it the time is a start time, and the second occurrence if the time is an end time. The end time/date will never represent a time instant preceding the start time/date. The list of times and dates will be terminated by a line consisting of the number 99.

Output will be one line for each line of input, and will be a single number (left justified) giving the number of times the hands cross between the starting time and date, and the ending time and date, taking note of the facts above. If the hands are together at the starting time or ending time, count that as a crossing.

EXAMPLE
Sample Input
```
00:00 15/09/1996  23:59  31/12/1996
02:00 15/09/1996  02:00  15/09/2000
01:00 13/04/1997  01:59  12/04/1998
02:01 09/04/2000  02:01 09/04/2084
99
```

Output for this Input
```
2376
32150
8013
675150
0
```