Problem A: Game

(blue balloon)

Four Quarters is a game of chance played with, well, four quarters. Two people, called A and B, each flip two quarters each round. They each gain or lose points each round based on the following tables:

		Player B		
		HH	HT	TT
Player A	HH	1	1	2
	HT	0	0	1
	TT	-1	0	0

Player A's payoff

		Player B		
		HH	HT	TT
Player A	HH	0	-1	-1
	HT	1	0	0
	TT	2	0	-1

Player B's payoff

There is no difference between Heads/Tails and Tails/Heads. As you can see, the odds are stacked in Player A's favor. At the beginning of the game, each player has 0 points, and points accumulate as the game progresses. At the end of the game, whichever player has the most points wins.

You must write a program that determines the probability that Player A will win, Player B will win, or they will tie, after a certain number of rounds. Assume that the coins are fair, i.e. that heads and tails are equally likely.

Standard Input

There is no input file for this problem.

Standard Output

Output a table that lists the probability that Player A will win, B will win, or they will tie, after each round for 1 to 20 rounds. The output for rounds 1 through 3 is given below.

Probabilities must be expressed as a percent, with 4 places after the decimal.

```
Round    A wins   B wins   Tie
1        43.7500% 18.7500% 37.5000%
2        56.6406% 22.2656% 21.0938%
3        62.3535% 22.7051% 14.9414%
```

Problem B: Prison Break

(red balloon)

A certain prison contains a long hall of ***n*** cells, each right next to each other. Each cell has a prisoner in it, and each cell is locked.

One night, the jailer gets bored and decides to play a game. For round 1 of the game, he takes a drink of whiskey, and then runs down the hall unlocking each cell. For round 2, he takes a drink of whiskey, and then runs down the hall locking every other cell (cells 2, 4, 6, …). For round 3, he takes a drink of whiskey, and then runs down the hall. He visits every third cell (cells 3, 6, 9, …). If the cell is locked, he unlocks it; if it is unlocked, he locks it. He repeats this for ***n*** rounds, takes a final drink, and passes out.

Some number of prisoners, possibly zero, realizes that their cells are unlocked and the jailer is incapacitated. They immediately escape.

Given the number of cells, determine how many prisoners escape jail.

Standard Input

The first line of input contains a single positive integer. This is the number of lines that follow. Each of the following lines contains a single integer between **5** and **100**, inclusive, which is the number of cells ***n***.

Standard Output

For each line, you must print out the number of prisoners that escape when the prison has ***n*** cells.

Sample Input	Sample Output
2 5 100	2 10

Problem C: TV game

(yellow balloon)

A new TV game show requires contestants to deduce a five letter word based on hints obtained by guessing other five letter words. The way the game is played is as follows: a secret five letter word is selected by the production staff of the game show. The object of the game is for the contestant to guess the secret word. The first letter of the secret word is revealed. The contestant will then guess a five letter word that may match the secret word. A computer then provides feedback to the contestant on the accuracy of the guess. Feedback consists of a report indicating if any letters in the guessed word are correct and in the same position in the secret word, if any letters in the guessed word are correct but not in the correct position in the secret word, and any letters in the guessed word that do not appear in the secret word.

As an example, the production staff chooses the secret word: “HELLO”. The contestant is told the first letter of the word is “H”. The contestant then guesses what the word could be, knowing it begins with the letter “H”. Let’s say the contestant guesses the word: “HOLES”. The game show computer would report that the “H” and “L” are in the secret word and in the correct position. In addition, the “O” and “E” are in the secret word, but in the incorrect position, and the “S” is not in the secret word. This is conveyed to the contestant by a single line report:

```
HoLe.
```

The upper case letters (“H” and “L”) indicate correct letter and position. The lower case letters (“o” and “e”) indicate correct letter, wrong position. The period (“.”) indicates a wrong letter (not in the secret word).

You will write a program that evaluates the contestant guesses, and prints out the single line report for each guess. If the contestant guesses the secret word exactly, then the five capital letters of the secret word will be displayed in the report.

Standard Input

The input data file consists of datasets for one or more games. A blank line marks the beginning of the next dataset (game). The line after the blank line contains the secret word. The remaining lines in the dataset represent the contestant’s guesses; there may be too few or too many guesses than are necessary to guess the secret word. The secret word will contain exactly five upper case letters. The contestant’s guesses, however, have to be checked for validity: valid guesses consist of exactly five upper case letters. Input is terminated by a dataset with the secret word: **“LINGO”** (that is, game play is stopped at that point, the program terminates, and no further guessing occurs).

Standard Output

Each game's output should be preceded by a single blank line (except for the terminating case). The first single line status report should be printed, which consists of the first letter of the secret word, followed by four periods. For each guess, print the single line status report for that guess. For an invalid guess, repeat the previous single line status report. If the guess exactly matches the secret word, that game ends and you should move on to the next one. The contestant may guess a maximum of six times; after the sixth guess, if the contestant did not guess the secret word, or you run out of guesses (the contestant gives up) print out the secret word in lower case letters and move on to the next game.

Sample Input	Sample Output
`HELLO`	`H....`
`HOLES`	`HoLe.`
`HAPPY`	`H....`
`HELMS`	`HEL..`
`HELLO`	`HELLO`
`HELPS`	
	`P....`
`PARTY`	`PAR..`
`PARKS`	`PAR..`
`PARES`	`PAR..`
`PARIS`	`PAR..`
`PONDER`	`PA.TY`
`PATTY`	`party`
`PUNTS`	
`PARTY`	`H....`
	`H....`
`HELIX`	`HELIX`
`HeLIX`	
`HELIX`	
`LINGO`	

Problem D: Conversion (Orange balloon)

Write a program to convert numbers in one base to numbers in a second base. There are 62 different digits:

{ 0-9,A-Z,a-z }

HINT: If you make a sequence of base conversions using the output of one conversion as the input to the next, when you get back to the original base, you should get the original number.

Standard Input

The first line of input contains a single positive integer. This is the number of lines that follow. Each of the following lines will have a (decimal) input base followed by a (decimal) output base followed by a number expressed in the input base. Both the input base and the output base will be in the range from 2 to 62. That is (in decimal) **A** = **10**, **B** = **11, …, Z** = **35**, **a** = **36**, **b** = **37**, …, **z** = **61** (**0-9** have their usual meanings).

Standard Output

The output of the program should consist of three lines of output for each base conversion performed. The first line should be the input base in decimal followed by a space then the input number (as given expressed in the input base). The second output line should be the output base followed by a space then the input number (as expressed in the output base). The third output line is blank.

Sample Input

```
8
62 2 abcdefghiz
10 16 1234567890123456789012345678901234567890
16 35 3A0C92075C0DBF3B8ACBC5F96CE3F0AD2
35 23 333YMHOUE8JPLT7OX6K9FYCQ8A
23 49 946B9AA02MI37E3D3MMJ4G7BL2F05
49 61 1VbDkSIMJL3JjRgAdlUfcaWj
61 5 dl9MDSWqwHjDnToKcsWE1S
5 10 42104444441001414401221302402201233340311104212022133030
```

Sample Output

```
62 abcdefghiz
2 1011100000100010111110010010110011111001001100011010010001

10 1234567890123456789012345678901234567890
16 3A0C92075C0DBF3B8ACBC5F96CE3F0AD2

16 3A0C92075C0DBF3B8ACBC5F96CE3F0AD2
35 333YMHOUE8JPLT7OX6K9FYCQ8A

35 333YMHOUE8JPLT7OX6K9FYCQ8A
23 946B9AA02MI37E3D3MMJ4G7BL2F05

23 946B9AA02MI37E3D3MMJ4G7BL2F05
49 1VbDkSIMJL3JjRgAdlUfcaWj

49 1VbDkSIMJL3JjRgAdlUfcaWj
61 dl9MDSWqwHjDnToKcsWE1S

61 dl9MDSWqwHjDnToKcsWE1S
5 421044444410014144012213024022012333403111042120221330 30

5 42104444441001414401221302402201233340311104212022133030
10 1234567890123456789012345678901234567890
```

Problem E : Decomposition

(purple balloon)

A sequence of positive integers is *Palindromic* if it reads the same forward and backward. For example:

```
23 11 15 1 37 37 1 15 11 23

1 1 2 3 4 7 7 10 7 7 4 3 2 1 1
```

A *Palindromic* sequence is *Unimodal Palindromic* if the values do not decrease up to the middle value and then (since the sequence is palindromic) do not increase from the middle to the end For example, the first example sequence above is **NOT** *Unimodal Palindromic* while the second example is.

A *Unimodal Palindromic* sequence is a *Unimodal Palindromic Decomposition* of an integer ***N***, if the sum of the integers in the sequence is ***N***. For example, all of the *Unimodal Palindromic Decompositions* of the first few integers are given below:

```
1:    (1)
2:    (2), (1 1)
3:    (3), (1 1 1)
4:    (4), (1 2 1), (2 2), (1 1 1 1)
5:    (5), (1 3 1), (1 1 1 1 1)
6:    (6), (1 4 1), (2 2 2), (1 1 2 1 1), (3 3),
         (1 2 2 1), (      1 1 1 1 1 1)
7:    (7), (1 5 1), (2 3 2), (1 1 3 1 1), (1 1 1 1 1 1 1)
8:    (8), (1 6 1), (2 4 2), (1 1 4 1 1), (1 2 2 2 1),
         (1 1 1 2 1 1 1), (      4 4), (1 3 3 1), (2 2 2 2),
         (1 1 2 2 1 1), (1 1 1 1 1 1 1 1)
```

Write a program, which computes the number of *Unimodal Palindromic Decompositions* of an integer.

Standard Input

Input consists of a sequence of positive integers, one per line ending with a **0** (zero) indicating the end.

Standard Output

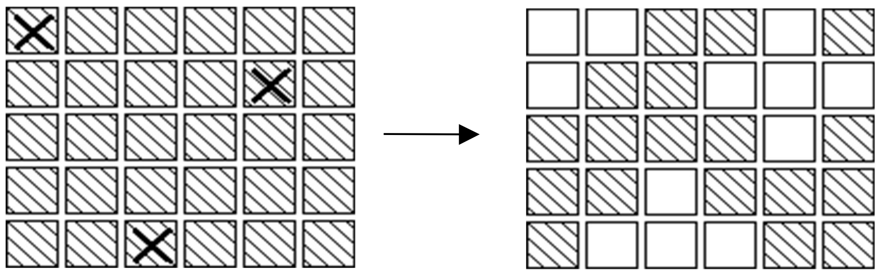
For each input value except the last, the output is a line containing the input value followed by a space, then the number of *Unimodal Palindromic Decompositions* of the input value. See the example on the next page.

Sample Input	Sample Output
2	2 2
3	3 2
4	4 4
5	5 3
6	6 7
7	7 5
8	8 11
10	10 17
23	23 104
24	24 199
131	131 5010688
213	213 1055852590
92	92 331143
0	

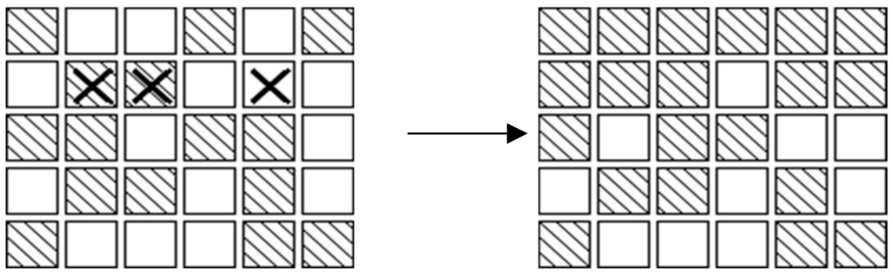
Problem F : Light Puzzle

(green balloon)

In an extended version of the game *Lights Out*®, is a puzzle with 5 rows of 6 buttons each (the actual puzzle has 5 rows of 5 buttons each). Each button has a light. When a button is pressed, that button and each of its (up to four) neighbors above, below, right and left, has the state of its light reversed. (If on, the light is turned off; if off, the light is turned on.) Buttons in the corners change the state of 3 buttons; buttons on an edge change the state of 4 buttons and other buttons change the state of 5. For example, if the buttons marked **X** on the left below were to be pressed, the display would change to the image on the right.



The aim of the game is, starting from any initial set of lights on in the display, to press buttons to get the display to a state where all lights are off. When adjacent buttons are pressed, the action of one button can undo the effect of another. For instance, in the display below, pressing buttons marked X in the left display results in the right display. Note that the buttons in row 2 column 3 and row 2 column 5 both change the state of the button in row 2 column 4, so that, in the end, its state is unchanged.



Note:

1. It does not matter what order the buttons are pressed.
2. If a button is pressed a second time, it exactly cancels the effect of the first press, so no button ever need be pressed more than once.
3. As illustrated in the second diagram, all the lights in the first row may be turned off, by pressing the corresponding buttons in the second row. By repeating this process in each row, all the lights in the first four rows may be turned out. Similarly, by pressing buttons in columns 2, 3 …, all lights in the first 5 columns

may be turned off.

Write a program to solve the puzzle.

Standard Input

The first line of the input is a positive integer ***n*** which is the number of puzzles that follow. Each puzzle will be five lines, each of which has six **0**'s or **1**'s separated by one or more spaces. A **0** indicates that the light is off, while a **1** indicates that the light is on initially.

Standard Output

For each puzzle, the output consists of a line with the string: "**PUZZLE #*m***", where ***m*** is the index of the puzzle in the input file. Following that line, is a puzzle-like display (in the same format as the input) . In this case, **1**'s indicate buttons that must be pressed to solve the puzzle, while **0**'s indicate buttons, which are not pressed. There should be exactly one space between each **0** or **1** in the output puzzle-like display.

Sample Input	Sample Output

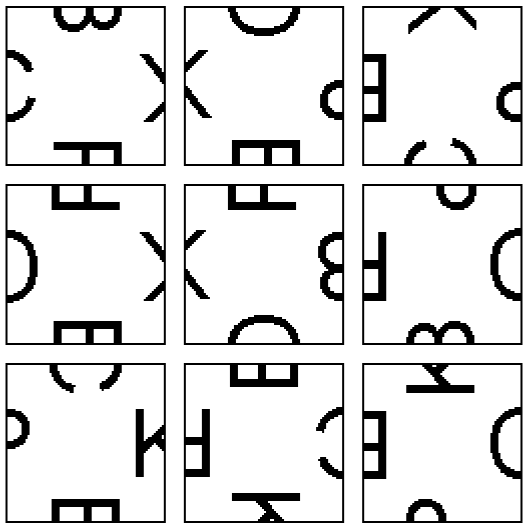
2
0 1 1 0 1 0
1 0 0 1 1 1
0 0 1 0 0 1
1 0 0 1 0 1
0 1 1 1 0 0
0 0 1 0 1 0
1 0 1 0 1 1
0 0 1 0 1 1
1 0 1 1 0 0
0 1 0 1 0 0
	PUZZLE #1
	1 0 1 0 0 1
	1 1 0 1 0 1
	0 0 1 0 1 1
	1 0 0 1 0 0
	0 1 0 0 0 0
	PUZZLE #2
	1 0 0 1 1 1
	1 1 0 0 0 0
	0 0 0 1 0 0
	1 1 0 1 0 1
	1 0 1 1 0 1

Problem G : Picture Puzzle

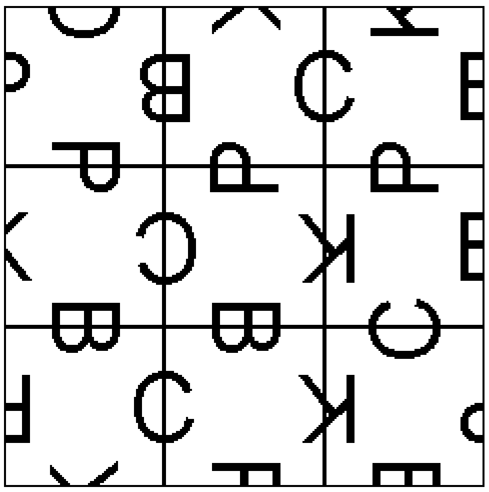
(white balloon)

One type of picture puzzle consists of nine square pieces, each of which has one-half of a picture on each edge. The pictures on each piece are either the left or the right half of one of four pictures designated B, C, K and P for this problem. The picture halves are aligned along the edges so that, if the left half is on one piece and the right on another, when the two pieces are aligned the pictures match. The purpose of the puzzle is to place the nine pieces into a three by three grid so that all the pictures along the adjacent edges match. Note that some of the pieces may need to be rotated to match.

Example puzzle:



Solution:



Write a program to solve one or more instances of the puzzle.

Standard Input

The input consists of a sequence of problems. Each problem begins with the problem number on a line by itself. The end of the data is indicated by a problem number of 0. Following the problem number line will be nine lines describing the pieces. Each of these lines begins with the piece number (1 through 9) followed by the picture on the top, right side, bottom and left side of the piece, in that order and separated by spaces. The picture halves are BL, BR, CL, CR, KL, KR, PL and PR. BL matches with BR, CL matches with CR, KL matches with KR and PL matches with PR. (For example, BL is the *left* half and BR is the *right* half of the picture designated B)

Standard Output

The output for each problem is to be: A line with the problem number followed by a colon (':').

If the problem has no solution, the next line should be “No Solution”. If there is a solution, that solution should be displayed as follows:

Since any solution may be rotated 90, 180 or 270 degrees to obtain another, the center square should be in the orientation given in the input and other squares aligned accordingly. Each row of pieces is displayed on three lines with a blank line between rows. The format for a single piece is:

<3 spaces><2 char top picture><3 spaces>
<2 char left picture><sp><1 digit piece number>><sp><2 char right picture><sp>
<3 spaces><2 char bottom picture><3 spaces>

A single blank line should follow the output for each problem.

Sample Input

```
1
1 BR KR PL CR
2 CL PR BL KR
3 KR PR CR BL
4 PL KR BL CL
5 PL BR CL KR
6 PR CL BR PL
7 CR KL BL PR
8 BL CR KL PL
9 KL CL PR BL
2
1 PR PR BL CR
2 BR KL CR PR
3 CR BL PL KR
4 KL PL BL CL
5 BR CR PL KR
6 KL BR PL CR
7 CL PL BL KR
8 KR KL CR BL
9 CR KL PR BL
3
1 PL KL CL BR
2 PL CR KL BL
3 PR BR CL KR
4 CR PL BR KR
5 PR CL BR KR
6 BL KL CR PR
7 PL BL CL KR
8 PR CR KL BR
9 KL BL CL PL
0
```

Sample Output

```
1:
   CL      KR      KL
PR 6 BR BL 2 CL CR 7 BL
   PL      PR      PR

   PR      PL      PL
KR 3 CR CL 4 KR KL 8 BL
   BL      BL      CR

   BR      BR      CL
PL 5 CL CR 1 KR KL 9 PR
   KR      PL      BL

2:
No Solution

3:
   KL      BR      BL
BL 6 CR CL 5 KR KL 9 CL
   PR      PR      PL

   PL      PL      PR
KR 7 BL BR 1 KL KR 3 BR
   CL      CL      CL

   CR      CR      CR
PL 2 KL KR 4 PL PR 8 KL
   BL      BR      BR
```