I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 25.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

6. Time of first submission will be used to break ties

II. Names of Problems

Number	Name
Problem 1	Balanced Words
Problem 2	Ball Drop
Problem 3	Birthday Finder
Problem 4	Clearing the Path
Problem 5	Cliff Climber
Problem 6	Complimentary
Problem 7	Fetch Me a Shrubbery!
Problem 8	Final Grade
Problem 9	Hangman
Problem 10	Hiking Challenge
Problem 11	Marble
Problem 12	Mutint
Problem 13	Neighbor
Problem 14	Rhonda
Problem 15	Rocket
Problem 16	Rounded Adder
Problem 17	Say Cheese
Problem 18	SDR
Problem 19	Semmy
Problem 20	Shape Rotation
Problem 21	Synonym Placer
Problem 22	The Bridge of Death
Problem 23	The Quest for the Holy Grail
Problem 24	Unique Paths
Problem 25	Vowels

1. Balanced Words

Program Name: BalancedWords.java **Input File: balancedwords.dat**

You have been asked to write a program that will count the number of balanced words in a text file. Words of odd length with odd ASCII totals are balanced and words of even length with even ASCII totals are balanced.

Input

There will be an unknown number of text lines as input. The text will not have any punctuation.

Output

Display the total amount of balanced words.

Example Input

```
This text is very random and has no point
It should be used for testing purposes only
I hope this problem goes well for your team
```

Example Output to Screen

```
The provided text has 13 balanced word(s).
```

2. Ball Drop

Program Name: BallDrop.java **Input File: balldrop.dat**

Your Physics teacher asked you to write a program that calculates how long it would take a ball to reach the ground when dropped from a given height, at an initial velocity of 0 meters per second.

The formula is:

$$time = \sqrt{2 * \frac{distance}{gravity}}$$

Gravity = 9.8 m/s^2

Input

There will be 4 inputs, each on their own line. Each input will be a single number representing the drop height in meters.

Output

For each input display the number of seconds it would take for the ball to hit the ground, rounded to 2 decimal places.

Example Input

```
548.2
172
51
9.8
```

Example Output to Screen

```
10.58 second(s)
5.92 second(s)
3.23 second(s)
1.41 second(s)
```

3. Birthday Finder

Program Name: BirthdayFinder.java **Input File: birthdayfinder.dat**

For fun you and your friends have decided to build a program that can determine someone's birthday given a date and their age in days on that date.

Input

The program will contain an unknown number of inputs, each on their own line.

Each input will be in the following format:

mm/dd/yyyy – daysOld

Output

For each input display its birthdate on its own line:

Each input will be in the following format:

mm/dd/yyyy

Example Input

```
05/30/2005 - 5
03/03/2005 - 368
7/01/1090 – 1582
```

Example Output to Screen

```
05/25/2005
02/29/2004
03/02/1086
```

4. Clearing the Path

Program Name: Clearing.java **Input File: clearing.dat**

You are in the process of building a game where the player tries to clear a path through rubble, using explosives. A single explosive can remove 1 rubble square in a level. Each level in the game will use a different rubble map and the number of explosives available to the player will vary map to map.

You want your game to have a lot of levels, so you created an algorithm for generating levels. You have found that some of your generated maps were unsolvable. To resolve these issues, you must write a second algorithm to verify that map is beatable with a given number of explosives.

Input

The input will include an unknown number of levels.

Each level will be the following format:

Bombs:MapWidth**x**MapHeight
MapGrid

The key for levels will be:

Terrain Type	Symbol
Start	S
End	E
Open	O
Rubble	R

Output

For each input display: Keep or Delete

- Display Keep when the map is solvable
- Display Delete when the map is not solvable

Example Input

```
5:7x1
SORORОE
3:7x1
SORRRRE
2:4x4
RRRR
RRER
RRRS
RRRR
1:4x4
ROOR
RRRR
RRRS
RERR
```

Example Output to Screen

```
Keep
Delete
Keep
Delete
```

5. Cliff Climber

Program Name: Climber.java **Input File: climber.dat**

You are preparing for a test run of grappling hook you have designed, with a releasable hook. You plan to use your new grappling hook to both climb up and repel down cliffs. Because the hook can be released you will only need one rope. The rope is heavy, so you only want the exact amount needed for your test site.

Each test site has an elevation map and you have decided to create a program that will read in an elevation map and tell you how much rope you should take. Your test run will always start at the top of the left most cliff. Each time there is an elevation change from that point on you will have to use your grappling hook to climb or repel some distance. After each repel/climb your rope is returned.

Input

The program will have a single input with an unknown number of lines. Each line will consist of hyphens that represent emptiness and C's that represent ground/cliff faces.

Note: The input will always be rectangular.

Output

Display the length of rope needed for the site.

Example Input

```
-------------------------------------------------
------C----------C--------------------------------
------C----------C--------------C-----------------
------C----------C--------------C------C----------
------C----------C--------------C------C----------
------C----------CC----C--------C------C----------
C-----C----------CC----C-----C--C--C---C----------
C-----C----C-----CC----C-----C--C--CC--C----------
C-----C----C-----CCCCCCCCC--CC--C--CCC-C---CCCCC--
C-----C----CC---CCCCCCCCCCCCCCCCC-CCCCCC---CCCCCC-
CCCCCCC----CC-C-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-
CCCCCCCC--CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-
CCCCCCCCC-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCC-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Example Output to Screen

```
10 units of rope are needed for this site.
```

6. Complimentary

Program Name: Comp.java Input File: comp.dat

Two's complement is used to express negative numbers in binary. For example, the value 30 in signed 8-bit binary is 00011110, and the signed 8-bit two's complement representation of -30 is 11100010. An easy way to convert from 00011110 to 11100010 is simply reverse 00011110 to become 11100001, and then add 1, which produces 11100010.

The difference between 30 and 18 is 12, which in 8-bit signed binary is 00001100. If you subtract 18 – 30, you get -12, which in two's complement (signed 8-bit) is the reverse of 00001100 + 1, or 11110011 + 1, or 11110100.

Your job is to read in two positive integers, express them in 8-bit signed binary, then their opposite values in two's complement, and then the difference between the two in 8-bit signed binary.

Input

Several pairs of positive integers X and Y, each pair on one line, with 0<X<=127 and 0<Y<=127.

Output

For each pair of values, five 8-bit signed strings, representing X, Y, -X, -Y, and X-Y, with a blank line following each output set.

Example Input

```
30 18
18 30
100 50
```

Example Output to Screen

```
30 = 00011110
18 = 00010010
-30 = 11100010
-18 = 11101110
12 = 00001100

18 = 00010010
30 = 00011110
-18 = 11101110
-30 = 11100010
-12 = 11110100

100 = 01100100
50 = 00110010
-100 = 10011100
-50 = 11001110
50 = 00110010
```

7. Fetch Me A Shrubbery!

Program Name: shrubbery.java Input File: shrubbery.dat

The Knights Who Say Ni have asked Sir Arthur to fetch them a shrubbery. He has given many qualifications. The Head of the Knights Who Say Ni have asked for a shrubbery that looks nice, and is not too expensive, and is as large as possible. There are many, many shrubberies in the world, and Sir Arthur doesn't have time to browse all of them.

Given the database of shrubberies, sort them based upon Sir Arthurs Request.

Remember he wants the LARGEST size, the SMALLEST price, and the LARGEST nice value.

Input

An integer N representing the number of data sets to follow. Each data set will have an integer Q representing the number of shrubberies to follow. Each shrubbery listing will have a string name followed by an integer rating of niceness, a decimal price, and decimal representing volume. At the end of the Q lines will be either the word "NICE", "PRICE", "SIZE" based upon which ever Sir Arthur is requesting.

Output

The shrubberies, by name, in order specified by Sir Arthur.

Assumptions

Sir Arthur can only request for shrubberies by size, price, or nice-ness. The number of shrubberies will not exceed 100. Decimal numbers will have no more than 5 digits after the decimal. There should be an empty line between data sets.

Example Input

```
6
3
Shrub1 4 5.0 7
Shrub2 5 6 6.2
Shrub3 1 7 5
SIZE
3
Shrub4 4 5 7
Shrub5 5 6 6
Shrub6 1 7 5
PRICE
3
Shrub7 4 5 7
Shrub8 5 6 6
Shrub9 1 7 5
NICE
```

Example Output to Screen

```
Shrub1
Shrub2
Shrub3

Shrub4
Shrub5
Shrub6

Shrub8
Shrub7
Shrub9
```

8. Final Grade

Program Name: FinalGrade.java **Input File: finalgrade.dat**

Your teachers have had enough…. They will no longer tell you what you need to make on your final to get an A in a course. The math is fairly simple and you have decided to write a program to calculate the final grade needed to receive an A in any class.

Your teachers do not round semester grades, so your semester grade must be 90.0 or higher to get an A. Your school has 3 grading periods per semester and one final. The grade you receive in a class is 25% of your final score plus 75% of the average of your 3 grading periods.

In equation form it looks like this:

SemesterGrade = GradingPeriodAverage*0.75 + FinalGrade*0.25

Input

The program will contain four inputs, each on their own line.

Each line of input will be the following format:

1st_Grading_Period, 2nd_Grading_Period, 3rd_Grading_Period

Output

Display the grade needed on the final to get an A, rounded to 2 decimal places.

Example Input

```
87.7, 48.23, 90.54
83.7, 94.2, 93.1
77.23, 98.65, 92.0
88.77, 87.44, 85.11
```

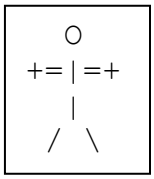
Example Output to Screen

```
133.53
89.00
92.12
98.68
```

9. Hangman

Program Name: Hangman.java Input File: hang.dat

In the classic blackboard game Hangman, you are given a word and attempt to guess the word by choosing a series of letters. In this program, you must determine if the provided letters are contained in the word prior to a body figure being drawn. One body part is added for each letter not contained in the word. The body figure below consists of the head shown by the letter "O", two body parts "|", 2 arms shown by equal signs "=", 2 hands shown by plus signs "+", and legs shown using a forward slash "/"and a back slash "\", 9 parts in all, drawn in order from left to right in four levels, as shown below. Once all the letters of the word are found, the game ends and no further letters are read. The game also ends once the figure is completed. If no body parts are drawn, print the word "SAFE".



Input

Several data sets, each on one line, consisting of a single word S, an integer N, followed by N letters.

Output

The word and resulting hangman figure, according to the series of guesses provided by the letters. The body parts MUST align as shown, with the "+" on the left edge of the screen, and all other parts aligned accordingly. At least one blank line must follow each output.

Example Input

```
APLUS 8 A S E U G P I J
UIL 9 A B C D E F G H j
SALTLICK 9 E F G H I J K M N
GOOGLE 4 G O L E
```

Example Output to Screen

```
APLUS
  O
+=|

UIL
  O
+=|=+
  |
 / \

SALTLICK
  O
+=|=+
  |

GOOGLE
SAFE
```

10. Hiking Challenge

Program Name: Hiking.java **Input File: hiking.dat**

You and your friends love hiking and enjoying covering ground quickly. You have decided it would be fun to have a challenge time for your hikes. You want to build a program that will generate tough, but achievable hiking time challenges.

You have broken many of your hiking maps into cells and assigned each cell as trail, wooded, overgrown or steep.

You have estimated the average time it should take to travel though each terrain as:

Terrain Type	Minutes to travel
Trail	3
Wooded	8
Overgrown	12
Steep	20

Now you need to build a program that analyzes terrain maps and gives you a challenge time for different hiking sites. You have decided your challenge times should be based on the fastest available path, not the shortest.

Input

The input file will consist of 3 inputs.

The first line of each input will contain your starting location and the destination, in this format:

{(startingColumn, startingRow), (endColumn, endRow)}

All the maps will be 5 by 5.

The key for the map is:

Terrain Type	Symbol
Trail	T
Wooded	W
Overgrown	O
Steep	S

Output

For each input display the goal time.

Example Input

```
{(0,0),(3,3)}
TTTTT
OSSSW
STOST
SSSSS
SWOWS
{(2,0),(2,3)}
TTTOO
TSSWO
TTOSO
TSSSO
TTTSS
{(0,1),(2,2)}
TTWTT
OSSTT
WSSTT
SSSTT
TTTTT
```

Example Output to Screen

```
Your time to beat is 46 minutes.
Your time to beat is 27 minutes.
Your time to beat is 35 minutes.
```

11. Marble

Program Name: marble.java **Input File: marble.dat**

There are 25 bins, each with a maximum capacity of 10 grams, into which we will sort marbles of weight ranging from 1 to 9 grams each. The basic idea is to sort marbles into the minimum number of bins, starting with bin 1, then using bin 2, etc. There will be 5 different strategies to use.

First Fit: Pick up the next marble. Scan the bins from 1 to 25 and put the marble into the first bin encountered that it will fit into.
First Fit Increasing: Sort all the marbles. Pick up the marbles from lightest to heaviest and apply the **First Fit** algorithm.
First Fit Decreasing: Same as **First Fit Increasing**, but pick up the marbles from heaviest to lightest.
Best Fit: Pick up the next marble from the original unsorted list. Look at all empty or partially filled bins. Put the marble into the bin that will come closest to maximum capacity. If more than one bin comes equally close to max capacity, then put it into the lowest numbered of those bins.
Worst Fit: Pick up the next marble of the original unsorted list. Look at all empty or partially filled bins. Put the marble into the bin that is the lightest. If more than one bin is equally light, then put it into the lowest numbered of those bins.

Overall Rules:
For all five strategies, the following rules take priority:

1) Since each bin costs money to use, you start using a new bin ONLY when it is impossible to fit the marble into any bin that is already started. Unused bins cost nothing.
2) A bin can never have more than 10 grams worth of marbles.

Input

Several sets of marbles. Each set lists the weights of the marbles, indicated by single digit positive integers all on one line, each separated by a single space.

Output

For each method, determine the weight of each bin. All used bins will have a weight > 0 and <=10, with no more than 25 bins used. Output only non-empty bins, with bin 1 first, 2 next, and so on in order.

Example Input

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

Example Output to Screen

```
FF: 10 9 8 9 7
FFI: 9 10 5 6 6 7
FFD: 10 10 10 10 3
BF: 10 9 8 9 7
WF: 9 9 9 9 7

FF: 10 10 9 10 10 8 4
FFI: 8 10 8 5 6 7 8 9
FFD: 10 10 10 10 10 9 2
BF: 9 10 10 10 10 8 4
WF: 10 9 9 7 8 10 8

FF: 10 10 10 9 9 10
FFI: 9 10 8 10 6 7 8
FFD: 10 10 10 10 10 8
BF: 10 10 10 9 9 10
WF: 9 9 9 9 8 7 7
```

12. Mutint

Program Name: Mutint.java **Input File: mutint.dat**

A “Mutint” is an integer M that is changed according to certain criteria, such as in this problem. Given a positive integer, change M according to the following rules.

1. Find the leftmost largest digit D of M.
2. If D is odd, change it to a zero.
3. If D is even, add 4 to that digit. If the sum exceeds 9, change D to the one’s place of the sum.

Input

Several integers, each on one line.

Output

M, according to the rules above.

Example Input

```
132
1421
18234
923
```

Example Output to Screen

```
102
1821
12234
23
```
(note: 023 is not correct)

13. Neighbor

Program Name: Neighbor.java **Input File: neighbor.dat**

Two fractions are considered "neighbors" if the numerator of the reduced positive difference between the two fractions is 1. For example, the fractions 1/11 and 1/12 subtract to be 1/132, and are therefore "neighbors". However, 2/5 and 4/5 differ by 2/5, and are not neighbors. Determine and output either the "neighbor" fraction value, or output "NOT NEIGHBORS".

Input

The positive integer values for two fractions, all on one line, in the order numerator, denominator, numerator, denominator, separated by single spaces.

Output

The positive "neighbor" difference between the two fractions, or the phrase "NOT NEIGHBORS".

Example Input

```
1 11 1 12
20 19 19 19
2 5 4 5
```

Example Output to Screen

```
1/132
1/19
NOT NEIGHBORS
```

14. Rhonda

Program Name: Rhonda.java **Input File: rhonda.dat**

Rhonda is printing processor chips, which can be made from fabricated components inscribed onto thin layers. As she prints the chips, Rhonda stacks two or more layers together to create a final chip made up of multiple layers.

She has several unique layers that can combined into various numbers of layers, which allows her to create a variety of chips. Each layered component is represented by a 10x10 grid of integers, with each cell containing an integer ranging from 0 to 9. The final chip is also 10x10 grid, with each cell being the sum of all the cells in that position for the layers that were used to create the chip.

For example, if Rhonda has fabricated 3 layers:

```
0123456789          1111111111          0011001100
0123456789          1111111111          1100110011
0123456789          1111111111          0011001100
0123456789          1111111111          1100110011
0123456789          1111111111          0011001100
0123456789          1111111111          1100110011
0123456789          1111111111          0011001100
0123456789          1111111111          1100110011
0123456789          1111111111          0011001100
0123456789          1111111111          1100110011
```

and decides to create a final chip that combines only the **first** and **second layers**, the resulting chip is represented by the 10x10 grid below:

```
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10
```

Each layer is indexed by the order given. The data set for this chip would be:

```
0 1
```

since this chip is made by combining the first and second layer in the order given.

Input

The first line of the input will an integer, `i`, representing the number of unique layers Rhonda has fabricated. This is followed by `i` sets of `10x10` grid of integers, each separated by a new line, representing the `i` fabricated layers. Following these will be several data sets, each a list of integers, all on one line, separated by spaces, which represent the layers added to create a chip.

Output

A `10x10` integer grid with each of the `10x10` cells represented by 2 digits and a space, the whole grid representing the final summed chip.

Assumptions

The cell values of each individual layer will be in the range 0-9, and the cell values of the final chip combined chips will in the range 0-99. All original fabricated layers will be unique.

Example Input

```
3
0101010101
0101010101
0101010101
0101010101
0101010101
0101010101
0101010101
0101010101
0101010101
0101010101

0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789

7386459728
3748244739
8420475820
1938475283
3847520734
5826752039
4230489329
4594890835
2188305727
4320834420

0 1
0 1 2
0 0 0 0 0 0 0 0 0 0
```

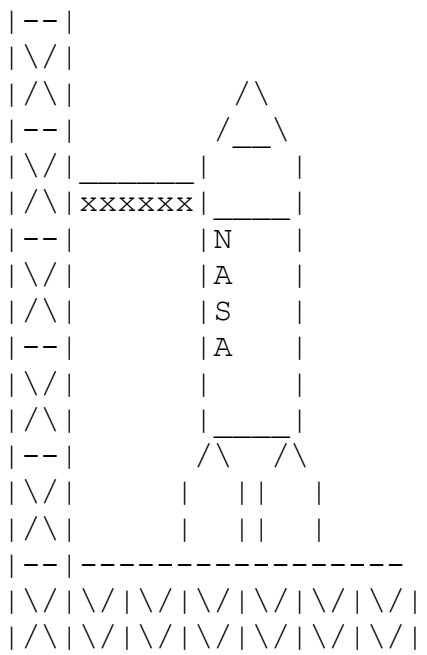
Example Output to Screen

```
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10
00 02 02 04 04 06 06 08 08 10

07 05 10 10 08 11 15 15 10 18
03 09 06 12 06 10 10 15 11 19
08 06 04 04 08 13 11 16 10 10
01 11 05 12 08 13 11 10 16 13
03 10 06 11 09 08 06 15 11 14
05 10 04 10 11 11 08 08 11 19
04 04 05 04 08 14 15 11 10 19
04 07 11 08 12 15 06 16 11 15
02 03 10 12 07 06 11 15 10 17
04 05 04 04 12 09 10 12 10 10

00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
00 10 00 10 00 10 00 10 00 10
```

15. Rocket

Program Name: Rocket.java

You are making a text-based space game and you want it display a rocket with a launch pad when the game loads. Write a program that displays the below ASCII art to the screen.



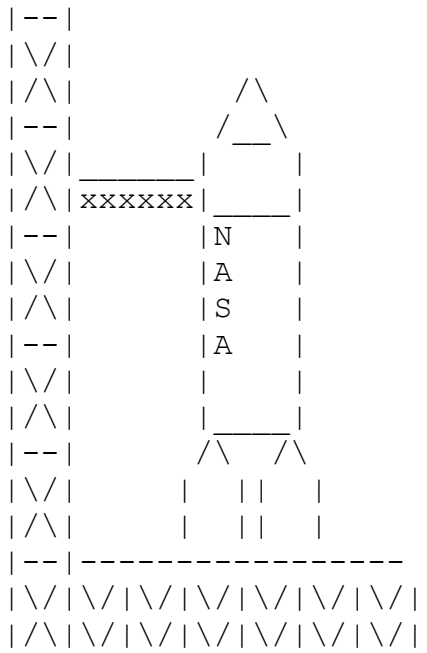
Input

None.

Output

Displays the rocket and launch pad.

Example Output to Screen



16. Rounded Adder

Program Name: RoundedAdder.java **Input File: roundedadder.dat**

A company has asked to write a program that adds rounded numbers together. They want the program to read in numbers, round them and then total their value.

Input

The first line will contain the number of decimal places needed. Then there will be an unknown number of values each on their own line.

Output

Display the total on the rounded number.

Example Input

```
2
7.5399
17.9355
2.9999
3.1888
.99968
.55777
48.65984
14.55447
78.16936
14.3568
```

Example Output to Screen

```
188.97
```

17. Say Cheese!

Program Name: Cheese.java **Input File: cheese.dat**

Satellites are abundant in earth's orbit, some designed to take high resolution photographs of our world. The launch day and time of these satellites is available, as well as the dates and times they would be over certain locations in the world, including your area. Given the launch time precise to the second, and a time the satellite would be directly over your location, you can calculate how long it will be from launch time to picture time, when you can look up into the sky and say, "Cheese!", to be a part of the satellite image!

For example, between a launch date of April 10, 2014 at 12:30:10 (military time), represented by the input string `041014123010`, and a "Say Cheese!" date of June 16, 2015, at 16:45:20, the total elapsed time is 432 days, 4 hours, 15, minutes and 10 seconds, represented by the output **`432:04:15:10.`**

Input

Several lines of data, with two values on each line, the first representing the "Say Cheese!" time for you, and the second the launch time of the satellite, each value in the form `MMDDYYhhmmss`, representing the Month, Day, Year, hour, minute, and second of the time.

Output

The span of time in days, hours, minutes, and seconds between the two given times. All times will be in the present millennium (year>2000). Assume also that all times are in standard time; no allowance for daylight savings times needs to be made. Output is to be in the format `dd:hh:mm:ss`.

Example Input

```
061615164520 041014123010
061602164520 042002205030
053015185045 022014164530
```

Example Output to Screen

```
432:04:15:10
56:19:54:50
464:02:05:15
```

18. SDR

Program Name: sdr.java **Input File: sdr.dat**

SDR stands for Split, Double, Reverse, which is what you are to do with each input string here. For each string, split it at the given integer, then output it in SDR format, as shown below. Study the output carefully, and you'll figure it out. Have fun!

Input

Several lines of data, each line containing a single capitalized word, followed by an integer guaranteed to be less than or equal to the length of the word.

Output

The SDR format of the given word and integer, as shown below.

Example Input

```
HOLIDAY 3
SNOW 2
RACECAR 4
TATTARRATTAT 6
```

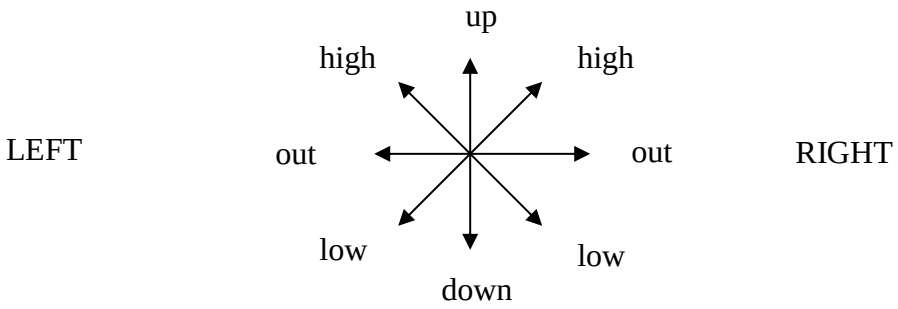
Example Output to Screen

```
HOL-YADI-LOH-IDAY
SN-WO-NS-OW
RACE-RAC-ECAR-CAR
TATTAR-TATTAR-RATTAT-RATTAT
```

19. Semmy

Program Name: semmy.java **Input File: semmy.dat**

In the semaphore signaling system, two flags are held, one in each hand, with arms extended, in various positions representing the letters of the alphabet. The pattern resembles a compass rose divided into eight positions: up (U), down (D), out (O), high (H) and low (L), for each hand. The left-hand signal is always read first. Six letters require a hand to be brought across the body so that both flags are on the same side. As an example, the letter H has the left hand across the body and held low (AL).



ALPHA	LEFT	RIGHT	ALPHA	LEFT	RIGHT
A	**D**	**L**	N	**L**	**L**
B	**D**	**O**	O	**AH**	**O**
C	**D**	**H**	P	**U**	**O**
D	**D**	**U**	Q	**H**	**O**
E	**H**	**D**	R	**O**	**O**
F	**O**	**D**	S	**L**	**O**
G	**L**	**D**	T	**U**	**H**
H	**AL**	**O**	U	**H**	**H**
I	**AL**	**U**	V	**L**	**U**
J	**O**	**U**	W	**O**	**AH**
K	**U**	**L**	X	**L**	**AH**
L	**H**	**L**	Y	**O**	**H**
M	**O**	**L**	Z	**O**	**AL**

Input

A data file that contains all letter/signal combinations on the first 26 lines, followed by two sets of data, each starting with an integer N. The first group contains N coded expressions that represents words or phrases, and the second group contains N English words or phrases. In the first group, a # indicates a space between words.

Output

Decode the first group of coded signal expressions into the English word or phrase, and encode the second group into the appropriate signal expression, with any spaces between words represented by the # sign.

Example input

```
ADL
BDO
CDH
DDU
EHD
FOD
GLD
HALO
IALU
JOU
KUL
LHL
MOL
NLL
OAHO
PUO
QHO
ROO
SLO
TUH
UHH
VLU
WOAH
XLAH
YOH
ZOAL
2
DLUOHLHHLO
HHALUHL#OOAHODHULLO
2
JAVA JIVE
FUN
```

Example Output to Screen

```
APLUS
UIL ROCKS
OUDLLUDL#OUALULUHD
ODHHLL
```

20. Shape Rotation

Program Name: ShapeRotation.java **Input File: shaperotation.dat**

You are working on building a game that is similar to Tetris and need to write and algorithm for rotating shapes. The bounds of the shapes will always be rectangle and the program will only need to rotate shapes clockwise by 0, 90, 180 or 270 degrees.

Input

There will be an unknown number of inputs. Each input will include dimensions for a shape, the rotation needed and finally the shape.

The first line of each input will define the bounds of the shape and what rotation is needed.

The format for this line will be:

width**x**height – degrees

The shape will be given over multiple lines. *'s will define the shape and –'s will be used for unused space.

Output

Display each rotated shape, separated by blank lines.

Example Input File

```
4x4 - 180
----
-*--
--**
--*-
6x2 - 90
******
--**--
3x3 - 0
---
*--
-*-
```

Example Output to Screen

```
-*--
**--
--*-
----

-*
-*
**
**
-*
-*

---
*--
-*-
```

21. Synonym Placer

Program Name: SynonymPlacer.java **Input File: synonymplacer.dat**

You keep getting marked off on your papers for over using words, so you have decided to write a program to make your life easier. The program will replace some of a given word with one of its synonyms.

Your program will take in a word, synonyms for that word and a chunk of text. The first time the word is found it will be left alone, the 2nd the word is found it will be replaced with the first synonym, the third time the word is found it will be replaced with the 2nd synonym and so on. After all the synonyms have been used the cycle will start back with original word.

Input

The first line of input will be a word followed by some of its synonyms, each separated by a comma.

Every line after the first will be text that needs to be processed by your algorithm.

Note: The number of synonyms and lines to be processed are unknown.

Output

Display the new text with the inserted synonyms.

Example Input File

```
good,wonderful,great,excellent
I like to get good grades. My parents like when I get good grades
and then I get to do more good activities. My favorite good activity
is playing on the computer. I am very good at computer games. I am
so good that I am ranked gold on LoL. Believe me, when I say that is
good.
```

Example Output to Screen

```
I like to get good grades. My parents like when I get wonderful grades
and then I get to do more great activities. My favorite excellent
activity
is playing on the computer. I am very good at computer games. I am
so wonderful that I am ranked gold on LoL. Believe me, when I say that
is great.
```

22. The Bridge of Death

Program Name: Bridge.java **Input File: bridge.dat**

If you do not know, when one is to cross the Bridge of Death, they must answer the Keeper, "these questions three." The Keeper gets to ask three questions, and if they do not know the answer, then they get launched into the pit. The traveler Sir Robin is up next. Your job is to write a program that will simulate in what scenarios Sir Robin will make it over the bridge. You will be given a dictionary for all of Sir Robin's knowledge. If the question asked of him by the Keeper is not in Sir Robin's dictionary, Sir Robin will be thrown into the pit. If all three questions are in Sir Robin knowledge, then Sir Robin will make it over the bridge.

Input

The first 50 lines will be the knowledge of Sir Robin, followed by an integer N representing the number of data sets to follow. Each data set will have 3 questions that the keeper asks Sir Robin.

Output

For each question, output the answer only if it is in Sir Robin's knowledge. If all three are answers are successfully output for each set of three questions, also output "Sir Robin can cross the bridge." At any point an answer is not in his knowledge, output "Sir Robin gets thrust into the pit." and proceed to the next set of questions. There should be a blank line between each output set.

Assumptions

Sir Robin will be the only traveler. Sir Robin's Knowledge will be of the form. "Question? Answer"

Example Input File

```
What is your name? Sir Robin
What is your favorite color? Red
What weighs the same as a duck? A witch
What is your quest? To seek the Holy Grail
Who is King of the Britons? Sir Arthur
Who keeps the Holy Hand Grenade? Brother Maynard
What is the frozen land? Nador
What is the surface are of a brick? 79cm squared
What does Karaoke mean in Japanese? empty orchestra
How many hours a day do cats sleep? 16 - 18 hours
What is the fear of teeth? Odontophobia
Who is the bravest Knight of them all? Sir Robin
What is the most money ever paid for a cow in auction? 1.3 million pounds
Which king of cards does not have a mustache? The King of Hearts
What is the only mammal that can't jump? Elephants
What textile are Australian $100 notes made of? plastic
What percentage of atoms in your body are replaced every year? 98%
Which do you burn more calories at sleeping or watching tv? sleeping
What was the first product to have a barcode? Wrigley's Gum
Which season do children grow fastest in? Spring
At what number are the measurements of Celsius and Fahrenheit equal? -40
How many toothpicks can be created from a cord of wood? 7.5 million
What are the things at the end of shoelaces called? Aglets
What was the occupation of Charlie Brown's father? A barber.
What is stressed spelled backwards? Desserts
```

What is the name of the tone that American car horns are in? F
What 3 western names are most known in China? Jesus Christ, Richard Nixon, and Elvis Presley
Who was the fastest public speaker in the world? President Kennedy
What was a bagpipe made of in Middle Ages? Skin of sheep
Which car gets more miles per gallon, stick or automatic? Stick
How many sesame seeds on average are there on a McDonalds Big Mac bun? 178
Who knows who built the Taj Mahal? No one
How far can a skunk spray? 10 feet
What family are Almonds a member of? Peach
What direction do bats turn when leaving a cave? left
How many eyelids do camels have? 3
What would Beethoven do when he sat to write music? pour water over his head
Which month in recorded history did not have a full moon? February 1865
What mammal has no vocal cords? Giraffes
What human bone is harder than concrete? Femur
Which side of a coin is more likely, heads or tails? tails
What country bans cheating on grades by law and jailtime? Bangladesh
Who invented scissors? Leonardo Di Vinci
What is Los Angeles's full name? El Pueblo de Nuestra Senora la Reina de los Angeles de Porciuncula
What American State has a one syllable name? Maine
What color of tooth brush is preferred most? Blue
Is non-dairy creamer flammable? Yes
What percentage of bones are in your feet? 25%
How long does it take the average person to fall asleep? 7 minutes
How many times per day does the average human laugh? 15 times
2
What is the name of the tone that American car horns are in?
What 3 western names are most known in China?
What color of tooth brush is preferred most?
What is the capital of Assyria?
What percentage of bones are in your feet?
What human bone is harder than concrete?

Example Output to Screen

F.
Jesus Christ, Richard Nixon, and Elvis Presley.
Blue.
Sir Robin can cross the bridge.

25%.
Sir Robin gets thrust into the pit.

23. The Quest for the Holy Grail

Program Name: Quest.java Input File: quest.dat

King Arthur is on a quest to find the Holy Grail. Obviously, he is having some trouble, and would like you to show him the path. However, there are some obstacles. King Arthur cannot traverse any location that is adjacent to a Killer Rabbit. He also cannot walk through canyons. All other spaces that are on the map can be traveled by King Arthur. Given a map with the location of the King Arthur, the obstacles, and the Holy Grail, show the shortest path to the grail with O's.

K – killer rabbit: King Arthur cannot walk here or any space adjacent
– canyon: King Arthur cannot walk here
. – empty space: King Arthur can walk here
A – King Arthur: King Arthur is currently standing here
H – the Holy Grail: King Arthur would like to be here.

Input

An integer N representing the number of data sets to follow and a 10x10 character matrix representing a map. Maps will be separated by a '-'.

Output

A 10x10 character matrix of the map with the highlighted shortest path.

Assumptions

There will always be a Holy Grail and a King Arthur located on the map. There will always be a path to the Holy Grail. Adjacent is defined in this problem as the 4 characters surrounding K (neither King Arthur nor the Killer Rabbit can move diagonally). There can be multiple shortest paths, as long as it as one of the shortest, it is correct. Note, there may not always be a rabbit on the map.

Example Input File

```
2
..........
..A.......
..........
.....K..#.
.........#.
######..#.
.........#.
.........#.
...H....#.
..........
-
......###.
..........
.###......
...#......
...#.....H
...#......
...K.#####
.........#
.#####...#
......A..#
```

Example Output to Screen

```
..........
..AOOOOO..
.......O..
.....K.O#.
......OO#.
######O.#.
...OOOO.#.
...O....#.
...H....#.
..........

......###.
OOOOOOOOOO
O###.....O
O..#.....O
O..#.....H
O..#......
O..K.#####
O.........#
O#####...#
OOOOOOA..#
```

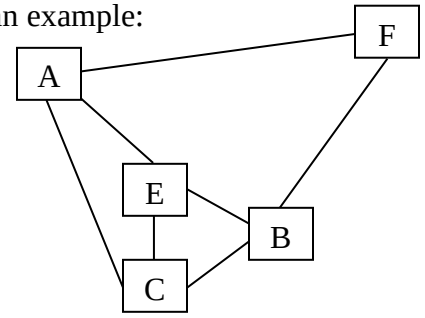
24. Unique Paths

Program Name: UniquePaths.java **Input File: uniquepaths.dat**

A road planner has come to you for help. He has a map of cities/roads and he wants you to write a program that will tell him how many unique paths there are from a given city to another. He is not concerned with the distance between the two cities, just how many unique paths there are from one city to another.

Two paths of the same length that contain the same cities are considered the same and should only be counted as single path.

Here is an example:



Notes:

- **A path cannot visit the same city more than once.**
- **The sample input listed is for the above map!**

The paths from A to F are: AF, AEBF, AECBF, ACBF, ACEBF.

There would be 4 unique paths, because AECBF and ACEBF visit the same cities and are of equal length.

Input

City names will always be letters ranging from A to Z.

The first line of input will be which cities you are trying to connect.

The format for the first line will be:
startingCity to endingCity

Next there will be an unknown number of lines listing the cities that are connected.

The format for connected cities:
CityA – CityB

Output

Display the number of unique paths in the following format:
The number of unique paths connecting the two cities is xxx.

Example Input File

```
A to F
A-E
A-C
E-C
C-B
E-B
B-F
A-F
```

Example Output to Screen

```
The number of unique paths connecting the two cities is 4.
```

25. Vowels

Program Name: vowles.java **Input File: vowels.dat**

In the language of the land of APLUS, the vowels are the letters A, P, L, U and S. To form the plural of a word, use the following rules:

1) If the word ends in a single consonant, add "XY".
2) If the word ends in a single vowel, drop that vowel and add "X"
3) If the word ends in a double consonant or a double vowel, add a second final letter, and then add "Y". For example, `PROBLEM` becomes `PROBLEMMY` because it ended in a double consonant.
4) If the word ends in more than two consecutive vowels or consonants, drop the leftmost of the consecutive vowels or consonants, and then add "Y". For example, `THEAPLUS` becomes `THEPLUSY` because it ends with five consecutive vowels.

Input

The first line will contain a single integer **n** that indicates the number of data sets that follow. Each data set will be a single word that will contain only the capital letters A through Z.

Output

The plural of each word according to the rules listed above.

Example Input File

```
6
PROBLEM
THEAPLUS
CAST
ADA
POLL
MASS
```

Example Output to Screen

```
PROBLEMMY
THEPLUSY
CASTXY
ADX
POLLLY
MSSY
```