2006 Consortium for Computing Sciences in Colleges
Programming Contest
Saturday, November 11th
Lipscomb University
Nashville, Tennessee

There are six (6) problems in this packet. Each team member should have a copy of the problems. These problems are NOT necessarily sorted by difficulty. You may solve them in any order. Each team will receive a bouncy ball in the color indicated below upon solving a problem correctly.

Remember input/output for the contest will be from `stdin` to `stdout`. `stderr` as well as any external file output will be ignored.

Have Fun & Good Luck! ☺

Problem 1. `valid` or `invalid`? *(blue ball)*

Problem 2. Binary Addition *(red ball)*

Problem 3. Can't Fool the Folio *(yellow ball)*

Problem 4. Bacon Numbers *(pink ball)*

Problem 5. Nutritional Analysis *(green ball)*

Problem 6. Reservations at the Opryland Resort *(peach ball)*

Problem 1

`valid` or `invalid`?

Suppose you have a programming language that defines a valid identifier as follows.

- An identifier must begin with an alphabetic letter, underscore (_), or a dollar sign ($).
- Subsequent characters in an identifier (if any) must be either alphabetic, digits, underscores, or dollar signs.
- A space cannot be embedded among the characters that form an identifier. A space is defined as a single blank character whose ascii value is 32.
- An identifier cannot be a reserved keyword used by the programming language. The case of letters in a keyword must match exactly for it to be reserved.

Your job is to write a program to determine whether a string represents a valid or invalid identifier using this definition.

Input

The input set starts with a listing of 1 or more reserved keywords, one per line, followed by a line starting with `LAST`. The string `LAST` should not be processed, and indicates the end of all keywords. This is followed by 1 or more strings, one per line, followed by a line starting with `END`. The string `END` should not be processed and represents the end of all input. No tab characters or other control characters will appear in the strings input.

Output

Your program should output each identifier read in, and indicate whether it is `valid` or `invalid` in the format shown below.

Sample Input

```
class
private
new
LAST
2x
R2D2
new
Joe's
cheese_soup
Main Street
Class
END
```

Output Corresponding to Sample Input

```
2x is invalid
R2D2 is valid
new is invalid
Joe's is invalid
cheese_soup is valid
Main Street is invalid
Class is valid
```

Problem 2

Binary Addition

Adding binary numbers is a very simple task, and very similar to the longhand addition of decimal numbers. As with decimal numbers, you start by adding the bits (digits) one column at a time, from right to left. Unlike decimal addition, there is little to memorize in the way of rules for the addition of binary bits:

```
0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 10
1 + 1 + 1 = 11
```

Just as with decimal addition, when the sum in one column is a two-bit (two-digit) number, the least significant figure is written as part of the total sum. The most significant figure is then "carried" to the next left column. Consider the following examples.

```
1001101 + 0010010 = 1011111
1001001 + 0011001 = 1100010
1000111 + 1010110 = 10011101
```

Input

The first line of input contains an integer *n*, (1 <= *n* <= 1000), which is the number of binary addition problems that follow. Each problem appears on a single line containing two binary values separated by a single space character. The maximum length of each binary value is 80 bits (binary digits). Note: The maximum length result could be 81 bits (binary digits).

Output

For each binary addition problem, print the problem number, a space, and the binary result of the addition. Extra leading zeroes must be omitted.

Sample Input

```
3
1001101 10010
1001001 11001
1000111 1010110
```

Output Corresponding to Sample Input

```
1 1011111
2 1100010
3 10011101
```

Problem 3

Can't Fool the Folio

A print shop has enlisted you to help them make some booklets.

Here is how booklets are made: an original manuscript consists of several 8½ x11 pages, and we need to photocopy these pages onto folios. A folio is a double-sized and double-sided sheet of paper (11 x 17) where we have two normal-sized pages on each side. Thus, one folio can accommodate 4 pages of information for the booklet.

After the pages of the manuscript are copied onto folios, the next step is to collate the folios and staple them (if necessary) into a booklet. To automate the procedure of making folios, the print shop needs to know the sequence of manuscript pages that appear on each folio.

For example, for a 4-page booklet, one folio will suffice. One side of the folio will contain pages 2 and 3, and the other side will contain pages 1 and 4. When this booklet is open you'll see page 2 on the left and page 3 on the right. If you flip the booklet over, you'll see page 4 on the left and page 1 on the right. Booklets with 5 or more pages will require at least 2 or more folios, in which case the folios need to be stapled together.

Input
The first line of input will give *n*, the number of booklets the print shop needs to make. The next *n* lines of input will contain the number of pages of each booklet we want.

Output
For each booklet, you need to output the order in which the pages should appear in the folios. The sequence of folios is such that when the booklet is open halfway through the pages, we simply have a stack of unfolded folios. You should output the folios' page numbers starting with the folio on top of the stack, and ending with the folio on the bottom.

For each folio, give the four page numbers in the following order:

- *left* page on the folio
- *right* page on the folio
- the page that appears on the *left* if we flip the folio over
- the page that appears on the *right* if we flip the folio over

If the number of pages in the manuscript is not a multiple of 4, then some of the pages in the booklet will be blank. Represent these blanks in your output with a hyphen. You may assume that the number of pages in a booklet is at least 1. Separate your page numbers with a single space, and always print 2 spaces after a colon. In your output, capitalize the words `Booklet` and `Folio`.

Sample Input

```
2 booklets
8 pages
10 pages
```

Output Corresponding to Sample Input

```
Booklet 1:
Folio 1:  4 5 6 3
Folio 2:  2 7 8 1

Booklet 2:
Folio 1:  6 7 8 5
Folio 2:  4 9 10 3
Folio 3:  2 - - 1
```

Problem 4

Bacon Numbers

Ever played "Six Degrees of Kevin Bacon?" The game involves determining how many steps someone is from actually appearing in a movie with Kevin Bacon. The minimum number of steps is known as the person's Bacon Number, which can be defined more precisely as follows:

The Bacon number of *x* is the smallest of:

- 0 if *x* is Kevin Bacon
- 1 + the Bacon number of someone who has appeared with *x* in a movie
- infinity

For example, the author of this programming contest problem has a Bacon number of infinity, since he/she has never appeared in a movie at all. On the other hand, the Bacon number of Adolf Hitler is 3:

- Kevin Bacon (BN 0) appeared in *Hero At Large* in 1980 with Kenneth Tobey (BN 1)
- Kenneth Tobey (BN 1) appeared in *The Great Sinner* in 1949 with Curt Bois (BN 2)
- Curt Bois (BN 2) appeared in a propaganda film in 1940 with Adolf Hitler (BN 3)
- There is no shorter chain from Kevin Bacon to Adolf Hitler

Your task is to determine a person's Bacon number.

Input

The input begins with an integer *n* (*0 < n < 500*) on a line by itself, which is the number of "actor pairs," followed by *n* actor pairs. Each actor pair represents two people who have appeared in a movie together. Each actor pair is on a line by itself, and always consists of four tokens each separated by one blank space. These four tokens will appear in the following order : the first name of the first actor, the last name of the first actor, the first name of the second actor, and the last name of the second actor.

Following the actor pairs, is a positive integer *k* on a line by itself, which is the number of queries. Each query consists of an actor name (first name and last name separated by a single space) on a line by itself. There will no extra spacing or blank lines in the input.

Output
For each query, print out the number of the query, the actor name, and the actor's Bacon number, in the exact format shown below.

Sample Input
```
3
Kevin Bacon Kenneth Tobey
Kenneth Tobey Curt Bois
Curt Bois Adolf Hitler
5
Adolf Hitler
Kevin Bacon
Kenneth Tobey
Curt Bois
Problem Writer
```

Output Corresponding to Sample Input
```
1: Adolf Hitler has a Bacon number of 3.
2: Kevin Bacon has a Bacon number of 0.
3: Kenneth Tobey has a Bacon number of 1.
4: Curt Bois has a Bacon number of 2.
5: Problem Writer has a Bacon number of infinity.
```

Problem 5

Nutritional Analysis

Manufacturers of food products are required to place nutrition information labels on their packages. A major part of this information is a listing of important vitamins and minerals, listing both the amount of the chemical present in one serving and the percentage of an adult's minimum daily requirement for that chemical.

Write a program to help prepare these nutritional labels by computing that percentage from the information on the amount present in one serving and the amount constituting the minimum daily requirement.

Input

Input consists of one or more lines, each of the form:

A U R V

where A is the amount of a vitamin/mineral present in one serving of the food product, U is the units in which A is measured, R is the minimum daily requirement for that vitamin/mineral, measured in the same units as A, and V is the name of that vitamin/mineral.

A and R will be floating point numbers. U will be a string of alphabetic characters with no embedded spaces. V will be a string of characters, possibly including spaces. A, U, R, and V will be separated from one another by exactly one space, and V is terminated by the end of the input line. End of the input is signaled by a line in which A is negative.

Output

For each line of input data, your program should determine the percentage of the recommended daily requirement being provided for that vitamin/mineral. If it is at least 1%, your program should print a line of the form

V A U P%

where V, A, and U are the quantities from the input, and P is the percentage of the minimum daily requirement represented by the amount A.

V should be printed left-justified on the line. A should be printed with 1 digit precision, and P with zero digits precision. V, A, U, and P should be separated by one space each.

After the last such line, your program should print a line stating

```
Provides no significant amount of:
```

followed by a list of the names of all vitamins/minerals which are provided as less than 1% of the minimum daily requirement. These should be printed one name per line, in the order they occurred within the input.

Sample Input

```
3500.0 iu 5000.0 Vitamin A
60.0 mg 60.0 Vitamin C
0.15 g 25.0 Fiber
109. mg 990. Phosphorus
0.0 mg 1000.0 Calcium
25.0 mg 20.0 Niacin
-1.0 x 0.0 x
```

Output Corresponding to Sample Input

```
Vitamin A 3500.0 iu 70%
Vitamin C 60.0 mg 100%
Phosphorus 109.0 mg 11%
Niacin 25.0 mg 125%
Provides no significant amount of:
Fiber
Calcium
```

Problem 6

Reservations at the Opryland Resort

Welcome to Music City USA! It was here in 1941 that static free radio was born after the city was granted the very first license to broadcast FM radio. Soon after, Nashville's fame skyrocketed as the Grand Ole Opry radio program began broadcasting live from the downtown Ryman Auditorium. It is now the longest running radio program in history.

Nashville prides itself on many fine dining establishments and hotel accommodations. Some of the most famous are at the Gaylord Opryland Resort & Convention Center, formerly known as the Opryland Hotel. Today it is the largest non-casino hotel in the world and bolsters the Nashville economy by attracting many trade shows and corporate meetings to the city. Their restaurants are so popular that many require reservations months in advance.

You have been hired to develop a calculator that can be used by guests to determine the earliest date when they can make reservations in many of the fine dining options available in the hotel. Guests are allowed to make reservations as far out as 180-days (approximately six months) in some restaurants, but only 90, 60, or 30 days out in others.

For example, if you desire to dine in the famous Cascades Restaurant, you are allowed to start making reservations as far out as 180 days in advance. So, for Thanksgiving Day this year (November 23rd), you could have started booking your meal on May 27th. At the Old Hickory Steak House for that day, 30 days is as far out as reservations are accepted, and reservations were not accepted until October 24th.

In order to fine tune your calculator, you have to remember that old adage you probably learned from your parents and elementary school teachers.

Thirty days hath September,
April, June, and November;
All the rest have thirty-one
Excepting February alone:
Which hath but twenty-eight, in fine,
Till Leap Year gives it twenty-nine.

The next leap year will be in 2008. So, we will need to make sure our calculator is built to take leap years into consideration. Recall a year is a leap year if it satisfies these conditions.

- Every year divisible by 4 is a leap year.
- Every year divisible by 100 is not a leap year, unless the year is also divisible by 400, in which case it is still a leap year.

Input

Your program will take as input an integer `n` representing the total reservations which will follow one reservation per line. Each reservation consists of the date the reservation is desired and an integer representing how many days out reservations are first available. Both are separated on a line by a single space. All dates are in the format `mm.dd.yyyy`. Two digits are used for all months and days (with a leading zero for months and days less than 10), and all years are shown using four digits. You may assume all dates are valid dates on the Gregorian 12-month calendar from the first day of 2005 through the end of 2054. Reservations are only checked for the periods of 30, 60, 90, or 180 days out.

Output

For each reservation in the input file, you will output the date in which reservations are first accepted exactly as shown in the format below.

Sample Input

```
10
05.15.2007 30
08.28.2007 30
06.28.2007 60
05.29.2007 90
02.28.2007 180
03.30.2007 180
02.28.2006 180
03.01.2006 180
01.30.2006 30
03.29.2008 30
```

Output Corresponding to Sample Input

```
30-day reservations for 05.15.2007 will be available on 04.15.2007
30-day reservations for 08.28.2007 will be available on 07.29.2007
60-day reservations for 06.28.2007 will be available on 04.29.2007
90-day reservations for 05.29.2007 will be available on 02.28.2007
180-day reservations for 02.28.2007 will be available on 09.01.2006
180-day reservations for 03.30.2007 will be available on 10.01.2006
180-day reservations for 02.28.2006 will be available on 09.01.2005
180-day reservations for 03.01.2006 will be available on 09.02.2005
30-day reservations for 01.30.2006 will be available on 12.31.2005
30-day reservations for 03.29.2008 will be available on 02.28.2008
```