Consortium for Computing Sciences in Colleges Southeastern Virtual Programming Competition Saturday, January 29th, 2022, 10 AM – 1 PM EST Hosted by Bob Jones University in Greenville, SC

There are nine (9) 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. Remember input/output for the contest will be from `stdin` to `stdout`. `Stderr` will be ignored. Do not refer to or use external files in your source code. Extra white space at the end of lines is ignored, but extra white space at the beginning or within text on a line is not ignored. Reminders for our competition:

- Each team may comprise 2 – 4 students, but only up to two students can be at a machine at a time.
- Internet access is only allowed to consult the APIs of a programming language. Cell phone usage is not allowed at any time.
- Any team found to be communicating with someone other than their teammates for assistance will be disqualified.
- Four programming languages are allowed: C#, C++, Java, Python
- An awards ceremony will be held at 1:30 PM EST.
- Have a lot of fun and good luck! 💻

Problem 1

French Country Names

French country names are feminine when they end with the letter e and masculine otherwise. The article “le” and a space are placed before masculine names, and “la” and a space precede feminine names. For example, “le Canada” and “la Belgique”. There are a few exceptions to this rule however. These five countries are masculine even though they end with e: Belize, Cambodge, Mexique, Mozambique, Zaire, and Zimbabwe. Also, if a country name starts with a vowel, use “l’ before it. That is, a lowercase letter l and an apostrophe. For example, “l’Afghanistan”. For the following plural country names, use “les”:

- les Etats-Unis
- les Pays-Bas

Your program should read in a French country name, and add the correct article before it. Note that a vowel is one of these five letters: a, e, i, o, and u.

Input
The first line of input will be a positive integer *n* representing the number of test cases. You may assume that $1 \le n \le 100$. This will be followed by *n* lines of input consisting of a country name. Assume the first letter of all countries is a capital letter, and all countries are one word spelled correctly in French.

Output Corresponding to Sample Input
The output should include the correct article placed appropriately with the test case as shown below.

Sample Input
```
6
Mexique
Canada
Etats-Unis
Australie
Bahrein
Gambie
```

Sample Output
```
le Mexique
le Canada
les Etats-Unis
l'Australie
le Bahrein
la Gambie
```

Problem 2

Hangman

In the game of Hangman, one player thinks of a word; the others try to guess what it is one letter at a time. The player draws a number of underscores equivalent to the length of the word. If a guessing player suggests a letter that occurs in the word, the other player fills in the blanks with that letter in the right places. If they suggest a letter that does not occur in the word, they get a strike. After four strikes, you lose. If you complete the word before four strikes, you win. Your job is to write a program to play this game.

Input

The first line of input will contain a single integer *n*, which represents the number of cases. You may assume that $1 \leq n \leq 100$. This will be followed by *n* test cases. Each test case consists of two lines. The first line contains a single word whose length is made up solely of capital letters. The second line will be a string of one or more letters separated by spaces representing the letters guessed. Assume all guesses are uppercase letters. A game always ends on the fourth strike or until they guess correctly. All games will be won or lost. Once a player wins, there will not be any extra guesses after the win.

Output

For each test case, you should indicate whether the game was won or lost in the format below. Include the case number, word, and the number of tries it took before a win/loss. If a loss, you should print a string with the underscore character _ replacing each letter not correctly selected. A blank line should precede each line of output.

Sample Input

```
3
COMPUTER
E R C M O Q P U Y T
COMPUTER
A B C D E F
SCIENCE
Z Y E C I S N
```

Output Corresponding to Sample Input

```

Case #1: Congrats, you won on COMPUTER in 10 tries

Case #2: Sorry, you lost on COMPUTER in 6 tries C_____E_

Case #3: Congrats, you won on SCIENCE in 7 tries
```

Problem 3

Multiply Your Money

In his spare time, Ronan has started to produce a new TV game show called *Multiply Your Money,* which is loosely based on the classic TV game show *Two for the Money*. Each contestant on the show is presented with three trivia questions. Each question has several possible correct answers. For example, a question might be, “Name the Hawaiian Islands.” The object of the game is for the contestant to come up with as many of the correct answers to each question as possible, because the buzzer will sound after a short period. The more correct answers the contestant provides the more prize money that can be won.

The contestant works on the questions one at a time. Here is how the contestant’s prize money is determined. If

A = the number of correct answers provided to the first question
B = the number of correct answers provided to the second question
C = the number of correct answers provided to the third question

then the contestant’s total prize money will be 5 * A * B * C dollars.

Here are some facts that you may assume about the game:

- Each of the three questions will have 5-10 possible correct answers.
- Each correct answer as well as each answer provided by a contestant will be a single word.
- The game will have 2-10 contestants, who all answer the same set of 3 questions in the same order. While one contestant is on stage, the other contestants are off stage, so they cannot see in advance, what the questions are.
- The contestant will provide 1-15 answers to each question.
- The contestant will not repeat an answer to each question.
- The correct answers are case sensitive and must be spelled correctly.
- The order of the correct answers does not matter. The contestant may provide answers in any order.
- Incorrect answers are ignored. There is no penalty for giving a wrong answer.

Input

The first three lines of input give all of the correct answers to the three trivia questions, with one question’s answers per line. The fourth line of input gives the number of contestants, n.
There will then be 4n additional lines of input, 4 lines of input per contestant. The format of these four lines will be as follows.

```
< single-word first name of the contestant >
< the contestant’s answers to the first question >
< the contestant’s answers to the second question >
< the contestant’s answers to the third question >
```

Output
The output will show the amount of prize money for each contestant. The contestants will appear in the output in the same order as they appeared in the input. There will be one line of output per contestant. Each line will have this format:

```
Prize money for < contestant’s name > = $ < integer >
```

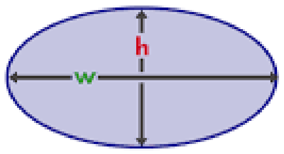
Please note that you need to have a space before and after the equals sign and a space after the dollar sign.

Sample Input
```
Niihau Kahoolawe Lanai Molokai Kauai Oahu Maui Hawaii
Chinese English French German Italian Japanese Latin Spanish
onion celery carrot potato beef salt pepper
2
Julia
Oahu Hawaii Maui Molokai Kauai
French Spanish Arabic Chinese
beef potato horseradish
Bob
Tahiti Samoa Guam Cuba Fiji Saipan
Spanish French German
water beef carrot celery onion
```

Output Corresponding to Sample Input
```
Prize money for Julia = $ 150
Prize money for Bob = $ 0
```

Problem 4

Area of Ellipse



This week, Melissa is learning about conic sections in her analytic geometry class. She would like you to write a program to help her study for her next test. One skill Melissa would like to master is to calculate the area of an ellipse when given its Cartesian equation in standard form.

The standard form of the equation of an ellipse is as follows.

$$Ax^2 + By^2 + Cx + Dy = E$$

where $A > 0$ and $B > 0$.

Here is the procedure for working out the area of the ellipse. By completing the square, the above equation can be converted into this form:

$$\frac{(x - x_1)^2}{p^2} + \frac{(y - y_1)^2}{q^2} = 1$$

Then, we conclude that the ellipse's center is at the point (x_1, y_1), and the area of the ellipse is πpq.

Input
The first line of input will contain the number of input cases, *n*. Each subsequent line of input pertains to one input case. An input case has five real numbers representing the constants A, B, C, D, E, respectively in the ellipse's standard form equation. You may assume there will be at least two ellipses but no more than 10.

Output
There will be one line of output per ellipse. Each line of output will be formatted as:
`Area of ellipse #` < *ellipse number* > = < *area* >

Where the ellipse number ranges from 1 to *n*. The area must be printed to exactly two decimal places of precision. There needs to be a space on both sides of the '#' sign, and on both sides of the '=' sign. You may assume that the ellipse's area does not exceed 999.99. The ellipse's area needs to be printed in a right-justified fashion so that the decimal point is exactly five characters to the right of the '=' sign.

If the coefficients of the equation of the ellipse describe a degenerate ellipse, in other words a graph that is a single point or no points at all, then the output line should have this format:
`Ellipse #` < *ellipse number* > `does not exist.`

Again, ensure that there is a space on both sides of the '#' sign.

Sample Input

```
4
0.4 0.6 0.8 -2.4 100
4 5 6 7 8
4 9 0 0 0
4 9 0 0 -1.5
```

Output Corresponding to Sample Input

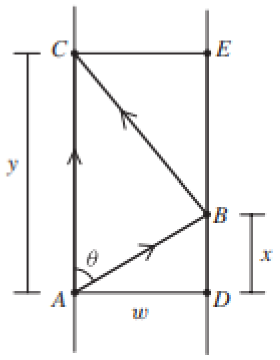
```
Area of ellipse # 1 = 659.23
Area of ellipse # 2 =   8.92
Ellipse # 3 does not exist.
Ellipse # 4 does not exist.
```

Problem 5

Walking with a Slower Friend

Fay and Sam decide to take a walk along their street. Fay's normal walking rate is v_f and Sam's is v_s, with $0 \leq v_s < v_f$. If Fay slows down and walks at Sam's normal rate, then they will be together for the entire trip, but Fay gets little exercise. In order for Fay to get her exercise while still being close to Sam, she has decided to walk at her normal rate diagonally across the street. That way, each time Fay returns, she meets with Sam who is walking parallel to the street. Of course, they want to maximize companionship, so Fay must walk at an angle that minimizes the time between meetings.

For example, Fay and Sam start together at *A* (see figure of one segment of a walking path below). Sam walks along the left side of the street, while Fay walks to a point *B* on the right side and then returns to meet Sam at *C*. Their entire walk then consists of repetitions of this segment.



Input
The first line of input will be the number of test cases, *T*. The next *T* lines will contain the width of the street *w* > 0 and two speeds v_s and v_f (constrained as above).

Output
For each test case, output the distance *y* Sam has traveled between meetings, and the time *t* between meetings. Round each number to four decimal places.

Sample Input
```
2
10.0 10.0 20.0
20.0 2.0 4.0
```

Output Corresponding to Sample Input
```
11.5470 1.1547
23.0940 11.5470
```

Problem 6

Stereographic Archery

Bernhard is practicing his archery in a large field. His aim is so good, that his arrows fire perfectly straight, seemingly ignoring gravity and air resistance. Bernhard is aiming at a target on the very top of a large dome in the center of the field. The dome is perfectly spherical, hollow, and has a unit radius. Since the arrows fire straight, he would not hit the very top, but rather some place lower on the dome.

Suppose we know the distance Bernhard fires from. Where on the dome is the arrow going to hit? Bernhard is also mathematically minded, and is curious where he would hit if he could travel an infinite distance out... could he actually hit the top of the dome then?

Input

The first line of input contains an integer t ($0 < t \leq 100$) denoting the number of test cases. Each test case will be a single line containing a distance d away from the center of the dome. The distance d will either be a real number ($-10^{12} \leq d \leq 10^{12}$) specified to three decimal places, or it will be `inf` or `-inf` denoting positive or negative infinity respectively.

Output

For each test case, output the case number, the distance d, and the height of the location the arrow hit, as formatted in the sample output. The distance d should be formatted to three decimal places, and the height should be rounded to six decimal places.

Sample Input

```
5
1.000
0.500
1.943
inf
-5.000
```

Output Corresponding to Sample Input

```
Case 1. At a distance of 1.000, the arrow hits at 0.000000
Case 2. At a distance of 0.500, the arrow hits at 1.000000
Case 3. At a distance of 1.943, the arrow hits at 0.581174
Case 4. At a distance of inf, the arrow hits at 1.000000
Case 5. At a distance of -5.000, the arrow hits at 0.923077
```

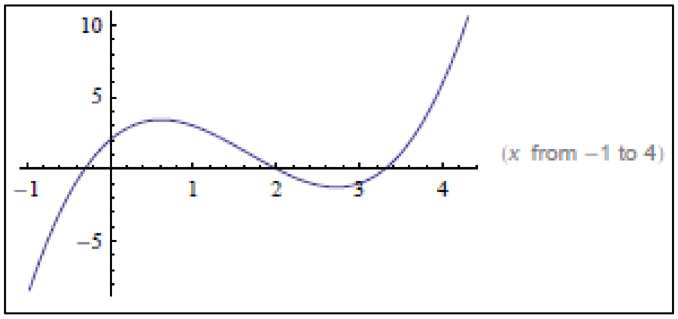
Problem 7

The Maximum-Minimum Theorem

We are in calculus class today learning about the Maximum-Minimum Theorem that states that given a continuous function *f* on a given interval [*a*, *b*], then *f* takes on both a maximum value *M* and a minimum value *m* on [*a*, *b*]. Your job is to write a computer program, which illustrates this theorem for the following function *f(x)* on any given interval [*a*, *b*].

$$\boxed{f(x) = x^3 - 5x^2 + 5x + 2}$$

Here is the graph of *f(x)* on the range from [-1, 4]. On this interval, we have a minimum of `-9` at `x = -1`, and a maximum of `6` at `x = 4`.



Input & Output

The first line of input contains an integer *T*, ($1 \leq T \leq 1000$), which is the number of test cases. This is followed by *T* lines, one line for each test case. Each test case contains two values each separated by a single space. The two values are two numbers *a* and *b* (in that order) representing your given interval [*a*, *b*] where *a* and *b* are both valid decimal point numbers, $-999{,}999.0 \leq a, b \leq 999{,}999.0$ and $a < b$. Your output will consist of *T* lines in the format demonstrated below. Round each floating-point number to the nearest tenth. Do not do any rounding until your output.

Sample Input

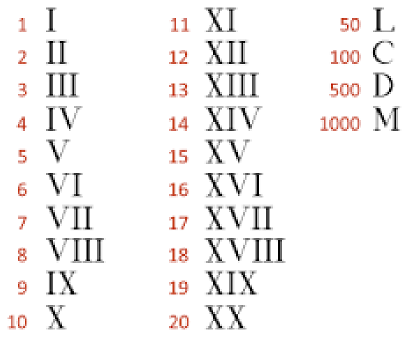
```
4
-1.0 4.0
-1.0 3.0
2.0 3.0
-10.0 10.0
```

Sample Output

```
case 1, minimum of -9.0 at x=-1.0, maximum of 6.0 at x=4.0
case 2, minimum of -9.0 at x=-1.0, maximum of 3.4 at x=0.6
case 3, minimum of -1.3 at x=2.7, maximum of 0.0 at x=2.0
case 4, minimum of -1548.0 at x=-10.0, maximum of 552.0 at x=10.0
```

Problem 8

Roman Numeral Summations



1	I	11	XI	50	L
2	II	12	XII	100	C
3	III	13	XIII	500	D
4	IV	14	XIV	1000	M
5	V	15	XV		
6	VI	16	XVI		
7	VII	17	XVII		
8	VIII	18	XVIII		
9	IX	19	XIX		
10	X	20	XX		

In this problem, you are going to create a Roman numeral calculator for doing summations. The Roman numerals you will need to comprehend are I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, and M = 1000.

To represent other values, these symbols, and multiples where necessary, are concatenated, with the smaller-valued symbols written further to the right. For example, the number 3 is represented as "III", and the value 73 is represented as "LXXIII". The exceptions to this rule occur for numbers having units values of 4 or 9, for tens values of 40 or 90, and for hundreds values of 400 and 900. For these cases, the Roman numeral representations are "IV" (4), "IX" (9), "XL" (40), "XC" (90), "CD" (400), and “CM” (900). Therefore, the Roman numeral representations for 24, 39, 49, 449 and 999 are "XXIV", "XXXIX", "XLIX", “CDXLIX” and "CMXCIX", respectively. Your calculator should display each input and the overall sum both as a Roman numeral and its equivalent value. Assume each number including the overall sum will be less than 4,000.

Input

The first line of input will be a positive integer *n* representing the number of locations. This is followed by *n* lines each with a valid Roman numeral.

Output

Output each Roman numeral input and its whole number equivalent in the format below. The overall sum should be displayed following a blank line on the last line as both a whole number and Roman numeral. No commas should be output.

Sample Input

```
6
IV
IX
XL
XC
CD
CM
```

Output Corresponding to Sample Input

```
Input Value #1 is IV = 4
Input Value #2 is IX = 9
Input Value #3 is XL = 40
Input Value #4 is XC = 90
Input Value #5 is CD = 400
Input Value #6 is CM = 900

Overall Sum is MCDXLIII = 1443
```

Problem 9

Image Processing

Pixel values for an image can range from 0 (black) through 255 (white) and are processed one at a time in row-major order. Your job is to write a problem which will decrease the pixel at coordinate `(row, column)` by the value at coordinate `(row + 2, column + 2)`, if such a coordinate exists. If the result of the subtraction is less than zero, the pixel is assigned the value 0. For example, consider the image matrix of pixel values below with four rows and five columns. Notice how the six values in the first two rows and three columns are modified, and all the other values remain unchanged.

Original Image Matrix

75	120	28	130	84
255	177	225	160	100
200	250	300	60	0
0	30	55	230	225

Image Matrix after Processing

0	60	28	130	84
200	0	0	160	100
200	250	300	60	0
0	30	55	230	225

Input

The first line of input will contain a single integer *N*, which represents the number of test cases. You may assume that $1 \leq N \leq 100$. This will be followed by *N* test cases. Each test case consists of a single line with a positive integer *R*, $1 \leq R \leq 100$ representing the number of rows, a single space, and a positive integer *C*, $1 \leq C \leq 100$ representing the number of columns. *R* rows and *C* columns of nonnegative integers of the image matrix of pixel values to be processed follow this. Each integer is separated by a single space. Your input will always be in row major order.

Output

For each image matrix of pixel values input, you should print out the new image matrix of pixel values after processing in the format as seen below. Include a blank line between each test case.

Sample Input

```
2
4 5
75 120 28 130 84
255 177 225 160 100
200 250 300 60 0
0 30 55 230 225
6 6
255 255 255 255 255 255
55  55  55  55  55  55
45  45  45  45  45  45
35  35  35  35  35  35
25  25  25  25  25  25
15  0   15   0  15    0
```

Output Corresponding to Sample Input

```
New Matrix 1:
0 60 28 130 84
200 0 0 160 100
200 250 300 60 0
0 30 55 230 225

New Matrix 2:
210 210 210 210 255 255
20 20 20 20 55 55
20 20 20 20 45 45
20 35 20 35 35 35
25 25 25 25 25 25
15 0 15 0 15 0
```