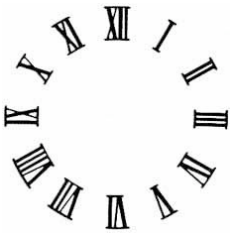
Baldwin-Wallace College

Spring 2006 Programming Contest

Do Not Open Until Instructed

Problem 1

Roman Numerals



Roman Numerals were developed in Ancient Rome and actually evolved from Etruscan numerals. The numerals were changed slightly in the Middle Ages into the system that lives on today. Nobody quite seems to know why we continue to use a numeral system that's a combination of base 2 and base 5, but you never know when you'll be building a monument and need the Roman Numeral version of a decimal number.

The Roman Numeral values are as follows:

I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, and M = 1000

Although there is a method for creating higher numbers, we are only considering those that may be legally formed using the above letters.

Roman Numerals other than those listed are formed according to the following rules (Courtesy of http://home.hiwaay.net/~lkseitz/math/roman/numerals.shtml):

I. The letters should be arranged from the one with the largest value to the one with the smallest. Each letter's value is added to the previous ones.
II. Only powers of ten (I, X, C, M) can be repeated. Do not repeat any letter more than three times in a row.
III. Because of the preceding rule, certain numbers must be written using subtraction. In this case, a letter with a smaller value precedes one with a larger value and the value of the smaller is subtracted from the larger. The result is then added to the rest of the letters. The following rules apply to subtraction:
    i. Only powers of ten (I, X, C, M) can be subtracted.
    ii. The smaller letter must be 1/5th (one-fifth) or 1/10th (one-tenth) the larger one.
    iii. The smaller letter must be either the first letter or preceded by a letter at least ten times greater than it.
    iv. If another letter follows the larger one, it must be smaller than the number preceding the larger one.

Note that the highest possible value that may be created is 3999 with our given letters and rules. Also, there is no symbol for 0 in this system. The goal of this problem is to convert given decimal integers into Roman Numerals if possible.

Details of the Input

The first line of the input contains a single integer, *n*, indicating the number of cases to follow. Each of the following *n* lines will contain an integer no greater than 10,000.

Details of the Output

If the integer falls in the legal range as defined above, print the Roman Numeral conversion. If the integer is illegal, output "Out of range."

Sample Input	*Sample Output*
3	VIII
8	XIV
14	Out of range.
99999	

Problem 2

Secure Data Sets

While preparing for the 3rd Annual High School Programming Contest, the Baldwin-Wallace College Contest Environment Development Team entered into a long debate about the need for keeping all the contest data sets encrypted. The team agreed that having the data sets encrypted would be beneficial and would keep out young hackers… but nobody really wanted to write up code to do the encryption. After several hours of tied rock-paper-scissors matches, they decided it would be easier to just turn the entire situation into a contest problem, which you are now reading.

The team did, however, come up with a relatively simple idea that unfortunately only works for alphabetic strings where all characters are already lower-case… but it's a start. The concept is simple – assign each character in the alphabet a value: a = 1, b = 2, c = 3 … z = 26. Moving character by character across the string, the current character shifts *k* letters forward in the alphabet where *k* is its assigned value as described above. The only exception is that when the current character is followed by a vowel in the original string, it moves forward *k* +1 letters in the alphabet. So, 'c' with a value of 3 would normally move three letters forward in the alphabet to become 'f,' unless it is part of a word like "cent" in which case it would move 4 letters forward to become 'g.' In this problem, the only vowels are a, e, i, o, u.

Your job is to write code to perform this encryption.

Details of the Input

The input will begin with a single positive integer, *n*, indicating the number of strings to follow, one per line. Each string will contain only lowercase alphabetic characters.

Details of the Output

For each input string, output the original string, followed by a space and the word "becomes" followed by another space and then the encrypted version of the string

Sample Input

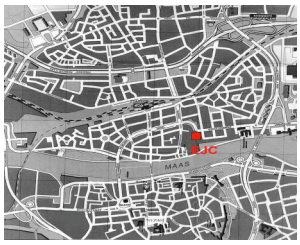
```
2
cent
programming
```

Sample Output

```
cent becomes gjbn
programming becomes fkdnkbzarbn
```

Problem 3

Campus Navigation



Steve is a math major at Cartesian College (a fictional school located in Northern Ohio). Originally from California, he is having trouble getting used to the snow (and therefore slush) that accumulates on the ground, even sometimes in early April. Steve has never owned a pair of boots, so he must figure out a way to get across campus by only walking on roads that have been plowed.

On one particularly unfortunate day, the snow was so deep that the College's Buildings & Grounds Crew was only able to plow two roads on campus. The crew then posted the equations of these roads on their website in y=*m*x + *b* form, where *m* is the slope of the road (when viewing the campus as a coordinate plane) and *b* is the y-intercept. All units are in feet.

Steve wants to write a program that takes the equations of the two roads and determines whether or not the roads intersect. If the roads do intersect, he would like to know where the intersection point is. Otherwise, he would like to know the square of the minimum distance he'll have to walk through the snow (since the numbers won't work out to nice integers if he goes for the exact distance).

Details of the Input

Each case in the input will consist of one line containing m_1 b_1 m_2 b_2, with m_1 and b_1 representing the slope and y-intercept of Road 1 and m_2 and b_2 representing the slope and y-intercept of Road 2. All values will be integers. The input will end when all four values are 0. Do NOT evaluate that case. [Hint: the Square of the distance between two parallel lines in terms of the values given is $(b_2-b_1)^2 / (m_2^2 + 1)$.]

Details of the Output

There are two possible outcomes – the roads intersecting or the roads being parallel. If the roads intersect, your output should report "The roads intersect at (*x*, *y*)" where *x* and *y* represent the coordinates where the roads intersect (they will always intersect at integer coordinates). If the roads are parallel, your output should report "The square root of *k* feet of snow must be crossed" where *k* is the minimum distance between the two roads.

Sample Input

```
3 0 3 10
2 0 -3 5
0 0 0 0
```

Sample Output

```
The square root of 10 feet of snow must be crossed
The roads intersect at (1, 2)
```

Problem 4

Topological Lite-Brite

Mathias is a 5 year old with above-average intelligence whose mother bought him a rectangular version of the popular (in the 1980's) children's game, "Lite-Brite." In the original Lite-Brite, a piece of black construction paper is illuminated from behind by a white light source, and the user inserts translucent colored pegs into designated slots to form a picture. Originally, the slots were aligned in a hexagonal pattern – but Mathias' mother knew a simple rectangular pattern would better serve his purpose.

As a student in the local university's Topology course, Mathias plays a game where he views the rectangular grid as a torus (a topological space basically equivalent to a donut) and draws geometric rays of different lengths onto the torus. **The only property of a rectangular representation of a torus that you will need for this problem is the following:** *If a ray extends beyond the edge of the rectangle, it continues in the same direction on the same row/column from the opposite edge.* (Although we're sure you'll enjoy looking up toruses when you get back to your high school!)

There is one more catch – the board starts completely black (due to the construction paper) with no pegs. As Mathias utilizes a slot, he always begins with red. When another ray intersects that slot, he removes the red peg and inserts an orange ray. He continues this process through Red, Orange, Yellow, Green, Blue, and Violet. After violet, he leaves the slot open again, but this time the construction paper has been broken, so white light shines through. He then continues again starting with red. To summarize, the color order goes Black (0 uses), Red (1 use), Orange (2 uses), Yellow, Green, Blue, Violet, White, Red, Orange, Yellow, Green, Blue, Violet, White, Red, Orange, etc…

Your task is to read in the rays drawn by Mathias and determine how many of the spaces on the board have been utilized and which color is the most prominent upon completion.

Details of the Input

Each case of the input will begin with one row of the form *r c n* indicating the number of rows (*r)* and columns (*c*) in the Lite-Brite (both positive integers no greater than 100), followed by the number (*n*) of rays Mathias intends to draw (no more than 500). Each of the following *n* lines will contain 4 values, *x y d l*. The values for *x* and *y* indicate the starting position of the ray (where x is the column counting from the left and y is the row counting from the bottom – the leftmost column is column 1 and the bottommost row is row 1) and *d* is the direction notated by the character 'U' indicating up, 'R' indicating right, 'L' indicating left or the character 'D' indicating down. The length of the ray, *l*, indicates how many pegs long the ray is including both endpoints. You may assume that

all (x, y) pairs will lie in the bounds of the Lite-Brite and that the length l will be a positive integer less than or equal to 1000. The input will terminate when $r = c = n = 0$.

Details of the Output

Each case's output will consist of 3 lines. The first line will consist of 6 *'s followed by a line which states "Torus k:" where k is the input case starting with 1. The next line will state "Used u/t spaces" where u is the number of slots on the grid that have been used at least once and t is the total number of slots on the grid. The third line will state "The color M appears z times" where M represents the most abundant color (capitalized) on the final grid. If more than one color is the most abundant, choose the first color in the order described above (including black and white). Do NOT insert a blank line between cases.

Sample Input (Diagram shown corresponds to the first case in the sample input)

```
3 3 3
3 2 R 2
3 1 U 2
2 2 D 3
5 5 1
5 4 R 3
0 0 0
```

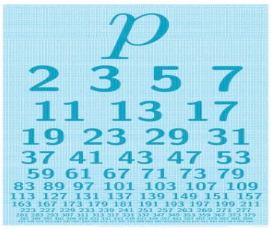
Sample 1	Column 1	Column 2	Column 3
Row 3	BLACK	RED	BLACK
Row 2	RED	RED	ORANGE
Row 1	BLACK	RED	RED

Sample Output

```
******
Torus 1:
Used 6/9 spaces
The color Red appears 5 times
******
Torus 2:
Used 3/25 spaces
The color Black appears 22 times
```

Problem 5

Goldbach



Goldbach's Conjecture states that all even numbers greater than 2 can be expressed as the sum of two prime numbers. Although it has never been proven (note: this could be your Ph.D. thesis in about 8 years – if that actually happens, please cite us as giving you the inspiration. Thanks.), it has been shown to hold for all evens up to 3×10^{17} by Oliveira e Silva… significantly larger than the bounds of the integer type which most programming languages recognize. For example, 20 can be written as both 3+17 and 7+13.

We also know by inspection that many odd numbers hold this same property. This problem is simple – for a given integer greater than 2, show every distinct way that integer can be expressed as the sum of two primes.

Primes are defined as integers which have exactly 2 distinct positive integers – itself and 1. 1 is not a prime because it can only be written as 1*1, and those are not distinct integers.

Details of the Input

Each input case will appear on its own line as a single integer, *k*, no greater than 10,000 but greater than 2. The input will terminate with the integer 0, which you should not evaluate.

Details of the Output

For each case, the first line of the output should be "*k*:" followed by *j* lines for each of the *j* ways that *k* can be uniquely expressed as the sum of primes. Each pair of primes should be in ascending order within the pair (3+17 as opposed to 17+3). If there is more than 1 way to express *k*, list them in ascending order by the first term (3+17 would come before 7+13). If there are no ways to do this, print "GOLDBACHED"

Sample Input	*Sample Output*
20 11 0	20: 3+17 7+13 11: GOLDBACHED

Problem 6

Determined Determinants

A *matrix* is simply a 2 dimensional collection of numbers. For example, $\begin{bmatrix} 2 & 4 \\ -3 & 5 \end{bmatrix}$ is a 2 x 2 matrix because it has 2 rows and 2 columns.

A determinant is a property of a *k* x *k* matrix that can be calculated many ways. One of those ways is by *cofactorization*.

First, the determinant of a 1 x 1 matrix is simply the value in the matrix. For a matrix A with dimensions *k* x *k* where k > 1, the determinant of A is

$$a(i1)A(i1) + a(i2)A(i2) + \ldots + a(ik)A(ik)$$

(*i* is typically the first row)

where

$a(ij)$ is the value i^{th} row and j^{th} column of A,

$A(ij) = (-1)^{(i+j)}$ * determinant of $M(ij)$

$M(ij)$ = the *k*-1 by *k*-1 matrix formed by deleting the i^{th} row and j^{th} column of A

Example: determinant of $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ = 1*((-1)$^{(1+1)}$ * deter. of [4]) + 2 * ((-1)$^{(1+2)}$ * deter. of [3])

= 1*(1*4) + 2*((-1)*3) = 4 + (-6) = **-2**.

Details of the Input

The first line of the input indicates the number of matrices that will follow. For each matrix, the first line contains one positive integer, *k* (no greater than 6), representing the number of rows/columns in the matrix. The following *k* lines will each contain *k* integers separated by spaces, representing the values in the matrix, one row represented per line.

Details of the Output

Print out the determinant of each matrix as calculated by cofactorization.

Sample Input

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

Sample Output

```
-2
10
```