University of Scranton
ACM Student Chapter / Computing Sciences Department
22nd Annual High School Programming Contest (2012)

Problem 1: Multiplication *à la russe*

The standard multiplication algorithm taught in elementary schools represents just one of several ways to compute the product of two numbers. A different method, which was known in ancient Egypt and is called multiplication *à la russe*, more closely resembles how multiplication is performed in electronic computers.

Using this method, we need not refer to (or memorize) the standard multiplication table (which, for any two single-digit numbers, tells us their product). Rather, we need to know how to do four things: add, decrement, double, and halve. Using these four operations, we can characterize multiplication of two positive integers a and b as follows:

$$a \cdot b = \begin{cases} b & \text{if } a = 1 \quad (1) \\ (a/2) \cdot 2b & \text{if } a > 1 \text{ and } a \text{ is even} \quad (2) \\ b \ + \ ((a-1)/2) \cdot 2b & \text{if } a > 1 \text{ and } a \text{ is odd} \quad (3) \end{cases}$$

To illustrate this, we multiply 214 by 3:

$$\begin{array}{lll} & 214 \cdot 3 & \\ = & 107 \cdot 6 & \text{by (2)} \\ = & 6 + 53 \cdot 12 & \text{by (3)} \\ = & 6 + 12 + 26 \cdot 24 & \text{by (3)} \\ = & 6 + 12 + 13 \cdot 48 & \text{by (2)} \\ = & 6 + 12 + 48 + 6 \cdot 96 & \text{by (3)} \\ = & 6 + 12 + 48 + 3 \cdot 192 & \text{by (2)} \\ = & 6 + 12 + 48 + 192 + 1 \cdot 384 & \text{by (3)} \\ = & 6 + 12 + 48 + 192 + 384 & \text{by (1)} \\ = & 642 & \text{by adding} \end{array}$$

Develop a program that, given two positive integers, multiplies them according to this method and outputs their product as the sum of numbers obtained during the process. For the example above, this sum is $6 + 12 + 48 + 192 + 384$.

Input: The first line contains a positive integer n indicating how many instances of the problem are described thereafter. Each of the next n lines contains a pair of positive integers a and b to be multiplied, separated by a space.

Output: For each pair a and b given as input, display on a single line the equation

$$a \ * \ b \ = \ t_1 \ + \ t_2 \ + \ \cdots \ + t_r$$

where the t_i's are the elements of the sum obtained by multiplying a and b using multiplication *à la russe*. The t_i's should be in ascending order.

```
Sample input
------------
4
214 3
1 3543
3543 1
134 18


Resultant output
----------------
214 * 3 = 6 + 12 + 48 + 192 + 384
1 * 3543 = 3543
3543 * 1 = 1 + 2 + 4 + 16 + 64 + 128 + 256 + 1024 + 2048
134 * 18 = 36 + 72 + 2304
```

Problem 2: Self-Divisible Numbers

A number is said to be *self-divisible* if each of its digits is a divisor of the prefix of the number up to and including itself. For example, 2136 is self-divisible because 2 is a divisor of 2, 1 is a divisor of 21, 3 is a divisor of 213, and 6 is a divisor of 2136.

Develop a program that, given a range of natural numbers (described by its lower and upper bounds), identifies all the self-divisible numbers in that range.

Note 1: The only number of which zero is a divisor is zero itself.

Note 2: Most programming languages include a so-called *modulo* operator by which to determine the remainder of a division. In Java, C, and C++, this operator is denoted by the percent sign, `%`. Thus, for example, the expression `13 % 5` yields 3 as its result.

Input: The first line contains a positive integer n indicating how many ranges of natural numbers are subsequently described. Each of the next n lines contains two natural numbers k and m, with $0 \leq k \leq m$, describing the range $k..m$.

Output: For each range given, the program displays a three-line message that identifies all self-divisible numbers in that range. The first line identifies the range, the second lists all the self-divisible numbers in that range, and the third is blank. (See sample output on next page for the intended format.) If there are numerous self-divisible numbers listed on the second line, it will "wrap around" when displayed. That's fine.

```
Sample input:
-------------
4
0 15
490 621
35670 35700
35 134


Resultant output:
-----------------
Self-divisible numbers between 0 and 15 are:
0 1 2 3 4 5 6 7 8 9 11 12 15

Self-divisible numbers between 490 and 621 are:
511 512 513 515 516 521 522 524 525 528 551 552 555 611 612 615 621

Self-divisible numbers between 35670 and 35700 are:


Self-divisible numbers between 35 and 134 are:
35 36 41 42 44 45 48 51 52 55 61 62 63 64 65 66 71 72 75 77 81 82 84 85 88
91 92 93 95 96 99 111 112 115 121 122 123 124 125 126 128
```

Problem 3: Reverse-Word Caesar Cipher Decoding

A **Caesar cipher** (named after the Roman Emperor Julius Caesar) is a rudimentary encryption scheme in which each letter is replaced by the letter occurring k places after it in the alphabet. (Imagine that the letters are listed in a circle so that **A** immediately follows **Z**.) The number k is referred to as the shift factor because, in effect, a Caesar cipher shifts all the letters of the alphabet k positions. (Non-letters are left unchanged.)

For example, if $k = 2$, **A** would be encoded as **C**, **B** would be encoded as **D**, ..., **X** would be encoded as **Z**, **Y** would be encoded as **A**, and **Z** would be encoded as **B**. The notion of a negative shift makes sense, too. For example, if $k = -3$, **A** would be encoded as **X**, **B** would be encoded as **Y**, **C** would be encoded as **Z**, **D** would be encoded as **A**, ..., and **Z** would be encoded as **W**. (Note that a shift of -3 is equivalent to a shift of 23; more generally, a shift of k is equivalent to a shift of $k + 26$.)

If, in addition, we encrypt a message so that each **word**'s characters are put into reverse order, we call that a Reverse-Word Caesar Cipher. By a word we mean a sequence of letters that

(a) either occurs at the beginning of a message or is immediately preceded by a non-letter, and
(b) either occurs at the end of a message or is immediately followed by a non-letter.

For example, using a shift factor of $k = 2$, the message **HELLO, WORLD!** would be encrypted as **QNNGJ, FNTQY!**

Develop a program that, given an integer k satisfying $-25 \leq k \leq 25$ and a message that has been encrypted using a Reverse-Word Caesar Cipher with a shift factor of k, decrypts the message and displays it. (You may assume that all letters appearing in the message are in upper case.)

Input: The first line contains a positive integer n indicating how many messages are to be decrypted. Following that are the encrypted messages, each occupying two lines, the first of which contains the shift factor k ($-25 \leq k \leq 25$) and the second of which contains the actual encrypted content of the message.

Output: For each given encrypted message, display the decrypted version on a line.

Sample input and output appear on the next page.

```
Sample input:
-------------
3
2
QNNGJ, FNTQY!
-5
YJJB AZDMB, TKJJIN
0
EREHW ERA UOY?


Resultant output:
-----------------
HELLO, WORLD!
GOOD GRIEF, SNOOPY.
WHERE ARE YOU?
```

Problem 4: Permutation Representation Conversion

A *permutation* of a set is a sequence that contains each member of that set exactly once. Here we shall be concerned with permutations of the set $\mathcal{Z}_m = \{0, 1, 2, \cdots, m-1\}$, where m is some positive integer. For example

$$\langle 4, 2, 5, 0, 1, 3 \rangle$$

is the permutation of $\mathcal{Z}_6$ in which 4 appears first, followed by 2, followed by 5, etc., etc.

Rather than simply listing the elements in the order in which they appear, an alternative way to describe a permutation is to indicate, for each of its elements, how many smaller elements precede it. That is, for a permutation P of the set $\mathcal{Z}_m$, and for each integer i satisfying $0 \leq i < m$, let k_i be the number of values smaller than i that precede i in P. The sequence $S_P = [k_0, k_1, k_2, \cdots, k_{m-1}]$ then uniquely describes P and hence is an alternative way of representing it. (We use square brackets here rather than angled brackets, as in describing P, simply to emphasize that the two kinds of sequences are interpreted differently.)

In our example permutation $P = \langle 4, 2, 5, 0, 1, 3 \rangle$, $k_5 = 2$ because two elements smaller than 5 (namely, 4 and 2) precede it and $k_2 = 0$ because no element smaller than 2 precedes it (due to both 0 and 1 coming later in the permutation). The reader should have no trouble calculating the other k_i's, thereby arriving at the sequence

$$S_P = [0, 1, 0, 3, 0, 2]$$

Develop a program that, given positive integer m and the sequence S_P, for some permutation P of the set $\mathcal{Z}_m$, outputs P.

Input: The first line contains a positive integer n indicating how many instances of the problem are to be solved. Each instance is described on two lines, the first of which contains a positive integer m and the second of which contains a sequence of nonnegative integers of length m corresponding to S_P, for some permutation P of the set $\mathcal{Z}_m$.

Output: For each sequence given, display it, followed by the word "represents", followed by the permutation that it represents. Use square brackets and angled brackets as shown in the sample output.

Hint: Place the largest number into the permutation, then the next-to-largest, etc., etc.

Sample input and output are shown on the next page.

```
Sample input:
-------------
2
6
0 1 0 3 0 2
13
0 1 1 3 4 0 2 6 1 5 0 8 8


Resultant output:
-----------------
[ 0 1 0 3 0 2 ] represents < 4 2 5 0 1 3 >
[ 0 1 1 3 4 0 2 6 1 5 0 8 8 ] represents < 10 5 8 0 6 2 9 1 12 11 3 7 4 >
```

Problem 5: Closest Point on a Line Segment

Develop a program that, given as input the cartesian coordinates of three points P, Q, and R, computes the coordinates of the point on line segment PQ that is closest to R.

Hint 1: The distance between points (x_1, y_1) and (x_2, y_2) is

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Hint 2: The slope of the line passing through points (x_1, y_1) and (x_2, y_2), assuming that $x_1 \neq x_2$, is $(y_2 - y_1)/(x_2 - x_1)$.

Hint 3: The slope-intercept form of the equation describing the line passing through point (x_1, y_1) and having slope m is $y = mx + b$, where $b = y_1 - (m_1 \cdot x_1)$.

Hint 4: For $i = 1, 2$, let (non-vertical) line L_i be given by the equation $y = m_i x + b_i$. Then, if their slopes are distinct, the x-coordinate of the point where the two lines intersect is given by $(b_2 - b_1)/(m_1 - m_2)$.

Input: The first line of input contains a positive integer n indicating how many instances of the problem are described thereafter. Each of the following n lines of input contains six real numbers, x_1, y_1, x_2, y_2, x_3, and y_3, separated by spaces. The intended interpretation is that point P is (x_1, y_1), point Q is (x_2, y_2), and point R is (x_3, y_3).

Output: For each given instance of the problem, generate three lines of output. The first identifies the endpoints of the given line segment PQ, the second identifies the given point R and the point S on PQ that is closest to R, and the third line is to be blank. See sample output below for the intended format and phrasing. As in the sample output, you may round off coordinates to the nearest thousandth, but this is not required.

Sample input and output appear on next page.

```
Sample input:
-------------
10
0.0 2.0 4.0 2.0 1.5 15.0
0.0 2.0 4.0 2.0 8.0 6.0
0.0 2.0 4.0 2.0 -2.0 0.0
1.0 -1.0 1.0 5.0 1.0 2.0
1.0 -1.0 1.0 5.0 5.0 3.0
1.0 -1.0 1.0 5.0 -2.0 -4.5
-2.0 1.0 4.0 -5.0 6.0 1.0
-2.0 1.0 4.0 -5.0 0.0 6.0
-2.0 1.0 4.0 -5.0 0.5 -1.5
7.0 5.0 -2.0 1.0 8.0 0.0


Resultant output:
-----------------
The point on segment [(0.0,2.0):(4.0,2.0)]
closest to point (1.50,15.0) is (1.5,2.0).

The point on segment [(0.0,2.0):(4.0,2.0)]
closest to point (8.0,6.0) is (4.0,2.0).

The point on segment [(0.0,2.0):(4.0,2.0)]
closest to point (-2.0,0.0) is (0.0,2.0).

The point on segment [(1.0,-1.0):(1.0,5.0)]
closest to point (1.0,2.0) is (1.0,2.0).

The point on segment [(1.0,-1.0):(1.0,5.0)]
closest to point (5.0,3.0) is (1.0,3.0).

The point on segment [(1.0,-1.0):(1.0,5.0)]
closest to point (-2.0,-4.5) is (1.0,-1.0).

The point on segment [(-2.0,1.0):(4.0,-5.0)]
closest to point (6.0,1.0) is (2.0,-3.0).

The point on segment [(-2.0,1.0):(4.0,-5.0)]
closest to point (0.0,6.0) is (-2.0,1.0).

The point on segment [(-2.0,1.0):(4.0,-5.0)]
closest to point (0.5,-1.5) is (0.5,-1.5).

The point on segment [(7.0,5.0):(-2.0,1.0)]
closest to point (8.0,0.0) is (5.979,4.546).
```

Problem 6: East Zordak's Congressional Districts

East Zordak is a new country. Much like the U.S., it has a legislative body each of whose members represents the residents of a particular congressional district within one of its member states.

Develop a program that, given the maps of the congressional districts of East Zordak's states, reports, for each district, not only its area but also the number of *contiguous regions* that compose it. (A definition of contiguous region appears below.)

Input: The first line contains a positive integer k equal to the number of states in East Zordak. The remaining input data are the descriptions of those states' congressional districts. The first line of each such description contains the name of the state. The second line contains a positive integer d indicating the number of congressional districts in that state, followed on the next line by two positive integers m and n that describe the north-south and east-west dimensions, respectively, of the state. (Conveniently for us, all states in East Zordak are rectangular in shape!). On the following m lines is the "map" of the state, which is given by an m-row $\times n$-column matrix, each element of which represents one square unit of land and indicates in which of the d congressional districts it lies. Each element is a positive integer in the range $1..d$; adjacent elements are separated by a single space.

Output: For each of the k states, the program should display its name and report, for each of its congressional districts, the district's area and the number of contiguous regions that compose it. (See the sample output below for the form in which this information is to be displayed.) Generate a blank line following the output for each state.

Definition of contiguous region: Consider a pair of square units of land, both in the same congressional district. They are defined to be in the same contiguous region of that district if and only if it is possible to get from one to the other (moving horizontally, vertically, or diagonally from one square to an adjacent square on each step) without ever entering a different congressional district.

Sample input and output appear on the next page.

Sample Input:

```
2
Zandar
6
7 10
4 4 4 4 2 2 1 1 3 3
4 4 4 2 2 2 1 1 3 3
4 4 2 2 2 1 1 6 5 5
2 2 3 3 3 3 6 5 5 5
2 2 3 3 1 3 5 5 3 5
2 2 2 3 3 4 4 5 3 5
4 4 4 5 5 3 3 3 3 3
Zork
5
7 20
4 4 4 4 2 2 1 4 4 4 1 1 1 5 5 5 5 5 5 5
4 4 4 2 2 2 1 4 4 1 1 1 1 1 1 5 5 5 3 3
4 4 2 2 2 2 2 1 1 1 1 1 1 5 5 5 3 3 3 3
4 4 2 2 2 2 2 5 5 5 5 5 5 3 3 3 3 3 3 3
2 2 2 5 5 5 5 2 5 5 5 3 3 3 3 3 3 3 3 3
4 2 2 5 5 5 5 2 2 2 2 3 3 3 3 3 3 1 1 1
4 4 4 5 5 5 5 2 2 2 2 2 3 3 1 1 1 2 2 2
```

Explanation:

East Zordak has two states.
This is name of first state.
It has 6 districts
and its size is 7-by-10.
This is the map showing in which
congressional district each
square unit of land lies.

Name of second state.
It has 5 districts
and its size is 7-by-20.

Resultant output:

```
In Zandar:
District 1 has area 7 and 2 region(s).
District 2 has area 15 and 1 region(s).
District 3 has area 20 and 2 region(s).
District 4 has area 14 and 3 region(s).
District 5 has area 12 and 2 region(s).
District 6 has area 2 and 1 region(s).

In Zork:
District 1 has area 23 and 2 region(s).
District 2 has area 33 and 2 region(s).
District 3 has area 30 and 1 region(s).
District 4 has area 20 and 3 region(s).
District 5 has area 34 and 1 region(s).
```