University of Scranton
ACM Student Chapter / Computing Sciences Department
20th Annual High School Programming Contest (2010)

Problem 1: Multiplying Products of Prime Powers

Each positive integer can be expressed (in a unique way, according to the Fundamental Theorem of Arithmetic) as a product of powers of the prime numbers. For example,

$$374556 \;=\; 2^2 \cdot 3^1 \cdot 5^0 \cdot 7^4 \cdot 11^0 \cdot 13^1 \cdot 17^0 \cdot 19^0 \cdot 23^0 \cdot 29^0 \cdot \;\cdots$$

To state it more precisely, for each positive integer m there exists a unique sequence $S(m) = \langle k_1, k_2, k_3, \cdots \rangle$ of natural numbers (i.e., nonnegative integers) such that

$$m = p_1^{k_1} \cdot p_2^{k_2} \cdot p_3^{k_3} \cdot \;\cdots$$

where p_i is the ith prime number (i.e., $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, etc.). Hence, to identify m, it suffices to describe the sequence $S(m)$. Using our example above, we get that

$$S(374556) \;=\; \langle 2, 1, 0, 4, 0, 1, 0, 0, 0, \cdots \rangle$$

A more economical description of the sequence would omit the trailing zeros (much as we routinely omit leading zeros from our numerals), yielding a sequence of length six: $\langle 2, 1, 0, 4, 0, 1 \rangle$.

Develop a program that, given the sequences $S(m)$ and $S(r)$ identifying positive integers m and r, respectively, calculates the sequence $S(mr)$ that identifies the product of m and r.

Input: The first line contains a positive integer n indicating how many instances of the problem are to be solved. Each instance of the problem is described by two sequences (corresponding to $S(m)$ and $S(r)$), one per line. Each sequence is given by a natural number t giving its length, followed by its t elements. For convenience, you may assume that $t \leq 20$.

Output: For each instance of the problem given as input, the program should output the two sequences ($S(m)$ and $S(r)$), separated by an asterisk (to indicate multiplication), followed by an equals sign and the sequence $S(mr)$. For formatting details, see the sample output below.

Sample input:

```
3
6 2 1 0 4 0 1
0
7 0 0 3 1 23 0 1
4 13 0 0 2
2 4 1
5 0 1 0 0 2
```

Resultant output:

```
<2 1 0 4 0 1> * <> = <2 1 0 4 0 1>
<0 0 3 1 23 0 1> * <13 0 0 2> = <13 0 3 3 23 0 1>
<4 1> * <0 1 0 0 2> = <4 2 0 0 2>
```

Problem 2: The $3n + 1$ Problem

Given a positive integer k, the $3n + 1$ sequence starting at k has as its first element k itself. For any particular element m in the sequence, where $m \neq 1$, the next element is either $3m + 1$ (if m is odd) or else $m/2$ (if m is even). The sequence ends with the first occurrence of 1.

For example, the $3n + 1$ sequence starting at 7 is

7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

This sequence has length 17.

The *Collatz conjecture* states that for every k, the $3n + 1$ sequence starting at k has finite length. Despite the efforts of many mathematicians over many years, this conjecture has yet to be proved or disproved.

Develop a program that, given a positive integer k, computes the length of the $3n+1$ sequence starting at k, as well as the largest value in that sequence.

Hint: If a and b are positive integers, the expression `a % b` (or `a mod b` in some programming languages) produces the remainder of the division of a by b. (E.g., `13 % 5` yields 3; `28 % 7` yields 0.)

Input: The first line contains a positive integer n indicating how many instances of the problem are described on the succeeding n lines. Each instance of the problem is described on a single line containing a single positive integer.

Output: For each positive integer k given as input, produce a message that identifies k, the length of the $3n + 1$ sequence starting at k, and the maximum value in that sequence. See the sample output below for proper formatting.

```
Sample input:      Resultant output:
------------       ----------------
4                  For 7, sequence length is 17 and maximum value is 52.
7                  For 40, sequence length is 9 and maximum value is 40.
40                 For 1, sequence length is 1 and maximum value is 1.
1                  For 3456, sequence length is 119 and maximum value is 9232.
3456
```

Problem 3: Polygon Area

A *polygon* is defined to be a plane figure formed by line segments such that

(1) each line segment intersects exactly two others, one at each endpoint, and
(2) no two line segments with a common endpoint are collinear (i.e., lie on the same line).

The endpoints of the line segments are referred to as the polygon's *vertices* (singular: *vertex*) and the line segments are referred to as its *sides*.

A polygon is said to be *convex* if, for any two vertices u and v, the line segment connecting them contains no points exterior to the polygon.

Below are two examples of polygons. The one on the left is convex; the one on the right is not, as is demonstrated by the dashed line segment, which connects two vertices and contains points exterior to the figure. For this problem, we are interested in convex polygons only.

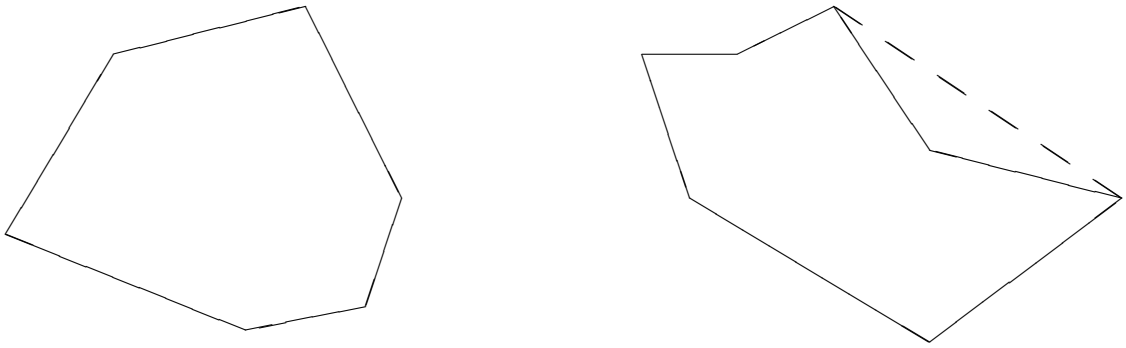

Figure 1: A Convex Polygon and a Non-convex Polygon

A standard way of representing a polygon is as a sequence $\langle p_1, p_2, \cdots, p_m \rangle$ of vertices, where, for each i satisfying $1 \leq i < m$, line segment $\overline{p_i p_{i+1}}$ forms a side of the polygon, as does $\overline{p_m p_1}$.

Develop a program that, given a convex polygon, computes its area.

Hint 1: By drawing some new edges in the right places (but always connecting vertices of the polygon), a convex polygon can be seen to be comprised of triangles. Hence, to compute the area of a convex polygon, it suffices to identify such a set of triangles and to sum their areas.

Hint 2: A triangle having sides of length a, b, and c has area

$$\sqrt{s(s-a)(s-b)(s-c)}$$

where $s = \frac{a+b+c}{2}$. This is known as *Heron's formula*.

Hint 3: The distance between points (x_1, y_1) and (x_2, y_2) is

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

Input: The first line contains a positive integer n indicating how many polygons are to be processed. Each polygon is described beginning with a positive integer $m > 2$ indicating how many vertices it has, followed by a sequence of vertices, one per line. Each vertex is given by two real numbers representing, respectively, its x- and y-coordinates.

Output: For each polygon provided as input, display its area on a line.

```
Sample input:
-------------
4
3
0.0 1.0
-1.0 0.0
1.0 0.0
4
-2.0 4.0
5.0 4.0
5.0 -1.0
-2.0 -1.0
3
1.0 4.0
7.5 2.6
-3.5 9.2
6
0.0 8.0
8.0 4.0
3.0 -1.0
-2.4 -1.5
-4.0 1.0
-4.0 5.0
```

```
Resultant output:
-----------------
1.0
35.0
13.75
73.65
```

Problem 4: Adjacent Repeated Substrings

In the realm of character strings, we use the term *square* to refer to any string that is of the form xx, where x is itself a string. That is, a square is composed of two copies of the same string, one after the other. (The term "square" comes from the fact that xx is often written as x^2.) For example, $abbabb = (abb)^2$

Consider the string $S = aabbabbababbabaa$. To explicitly show the positions at which the characters occur, here is another depiction of S:

```
  0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| a | a | b | b | a | b | b | a | b | a | b | b | a | b | a | a |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Notice that positions are numbered starting at zero.

Embedded within S are several substrings that are squares. For example, $S[1..6] = (abb)^2$ and $S[3..12] = (babba)^2$.

Develop a program that, given a string of characters, finds all substrings that are squares of length at least six.

Input The first line contains a positive integer n indicating how many subsequent character strings are provided as input. Those strings appear on the next n lines, one per line. Each string is composed of letters.

Output For each character string given as input, display the string on one line and then, on subsequent lines, identify each of its substrings that are squares of length at least six, one per line. To identify a square, display its starting position, followed by a space, followed by the first half of the square. The squares should be listed in decreasing order by length. For two squares of the same length, they should be listed in ascending order by starting position. Following the last square listed, put a blank line.

Sample input and output is on the next page.

```
Sample input
------------
3
abaabaaaba
aabaaab
aabbabbababbabaa



Resultant output
----------------
abaabaaaba
2 aaba
0 aba
1 baa

aabaaab

aabbabbababbabaa
3 babba
4 abbab
5 bbaba
1 abb
2 bba
3 bab
8 bab
```

Problem 5: Normal Form Polynomials in Plain Text

A *polynomial* is an expression of the form

$$c_k x^k \ + \ c_{k-1} x^{k-1} \ + \ c_{k-2} x^{k-2} \ + \ \cdots \ + \ c_2 x^2 \ + \ c_1 x^1 \ + \ c_0 x^0$$

where k (called the *degree* of the polynomial[1]) is a nonnegative integer and the c_i's (called the *coefficients*), are real numbers. Each subexpression of the form $c_i x^i$ is called a *term*. The first term, $c_k x^k$, is called the *leading term*, and the last one, $c_0 x^0$, is called the *constant term*. (Note that $c_0 x^0 = c_0 \cdot 1 = c_0$, so the value of the last term does not depend upon x and hence is a constant.)

Here we shall be concerned only with polynomials having integer coefficients. An example of such a polynomial is

$$-5x^5 \ + \ 1x^4 \ + \ 0x^3 \ + \ 13x^2 \ + \ -3x^1 \ + \ 4x^0$$

If this looks odd to you, it's because in the standard way of writing a polynomial, we
(a) omit x^0 from the constant term
(b) omit the exponent 1 (in x^1)
(c) omit any coefficient 1 (except if it is in the constant term)
(d) omit any term whose coefficient is zero (unless it is the only term!)
(e) turn any negative coefficient (unless it is in the leading term) into a positive one by replacing the binary plus sign preceding it by a binary minus sign.

A polynomial written in this way is said to be in *normal form*. For example, the normal form of the polynomial above is

$$-5x^5 \ + \ x^4 \ + \ 13x^2 \ - \ 3x \ + \ 4$$

Develop a program that, given the coefficients of a polynomial, displays the normal form of that polynomial. Because of the limitations of "plain text", there is no really nice way of displaying exponents. The best we can do is to display a polynomial on two lines, with the exponents on the first line and the rest of the symbols, lined up properly with the exponents, on the second line, as in

```
   5    4       2
-5x  + x  + 13x  - 3x + 4
```

[1]Technically, for $k > 0$ to be the degree requires that $c_k \neq 0$, but we will not worry about that here.

Input: The first line contains a positive integer n indicating how many polynomials are described on succeeding lines. Each polynomial is described on two lines, the first of which contains its degree k and the second of which contains its $k+1$ coefficients (beginning with the coefficient of its leading term and ending with the coefficient of its constant term).

Output: For each polynomial given as input, display it in normal form, on two lines, as described above and as exemplified in the sample output below, followed by a blank line. In particular, each binary operator (plus or minus) separating adjacent terms should be placed so that exactly one column to its left and one column to its right are blank (in both rows).

```
Sample input:
------------
2
5
-5 1 0 13 -3 4
14
15 0 0 -3 0 0 0 0 0 0 0 32 0 2 -4


Resultant output:
----------------
   5    4      2
-5x  + x  + 13x  - 3x + 4

   14     11      3
15x   - 3x   + 32x  + 2x - 4
```

Problem 6: Base k Addition

Most peoples of the world use the *decimal* (or *base ten*) numeral system, in which the ten symbols 0, 1, 2, ..., 9, called the decimal *digits*, are used for forming numerals. The contribution made by each digit in a numeral depends not only upon its value (e.g., 5 vs. 7) but also upon its position within the numeral. The rightmost position is the 1's column; moving to the left from there we encounter the 10's, 100's, 1000's, etc., columns. Note that these are the powers of ten. For example, the decimal numeral 7204 corresponds to the sum

$$7204_{10} \;=\; (7 \cdot 10^3) + (2 \cdot 10^2) + (0 \cdot 10^1) + (4 \cdot 10^0)$$

(Notice that, to indicate a numeral's base explicitly, we put it as a subscript to the right of the numeral.)

There is nothing special about using 10 as the base, however. In a base 5 numeral, for example, only the digits 0 through 4 may appear, and in such a numeral we find the 1's, 5's, 25's, 125's, etc., columns, corresponding to the powers of five. Hence, the base 5 numeral 3042 represents the number given by the sum

$$(3 \cdot 5^3) + (0 \cdot 5^2) + (4 \cdot 5^1) + (2 \cdot 5^0)$$

in a manner completely analogous to the decimal numeral 7204 above.

Performing addition in base 5 (or in any other base) is completely analogous to how it is done in base 10, too. For example, in base 5 we have

```
  44032
+  1341
------
100423
```

In the 1's column, we have $2_5 + 1_5 = 2_{10} + 1_{10} = 3_{10} = 3_5$, so we record a 3 in that column. In the 5's column, we have $3_5 + 4_5 = 3_{10} + 4_{10} = 7_{10} = 12_5$, so we record 2 in that column and carry the 1 to the 25's column. In the 25's column, we have (remembering the incoming carry) $1_5 + 0_5 + 3_5 = 1_{10} + 0_{10} + 3_{10} = 4_{10} = 4_5$, so we record 4 there. In the 125's column, we have $4_5 + 1_5 = 4_{10} + 1_{10} = 5_{10} = 10_5$, so we record 0 and carry the 1. And so on.

Develop a program that, given as input an integer k, with $2 \leq k \leq 9$, and two base k numerals, calculates their sum and expresses it as a base k numeral.

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 three lines, the first of which contains the base and the next two of which contain two numerals of that base, one per line, each having at most thirty digits.

Warning: The judges' test data will include numerals representing numbers that are not in the range of values covered by intrinsic data types (e.g., `int`, `long`, `float`, `Integer`, `Real`) in languages such as C, C++, Java, or Pascal.

Output: For each pair of numerals to be added, generate a single line of output that identifies their base, the two numerals (separated by a plus sign), an equals sign, and their sum. (See the sample output below.)

```
Sample input:          Resultant output:
-------------          -----------------
4                      In base 10, 33468294890 + 7502 = 33468302392
10                     In base 2, 111110001 + 110100101010011 = 110101101000100
33468294890            In base 6, 543052 + 242441 = 1225533
7502                   In base 9, 543052 + 242441 = 785503
2
111110001
110100101010011
6
543052
242441
9
543052
242441
```