University of Scranton
ACM Student Chapter / Computing Sciences Department
26th Annual High School Programming Contest (2016)

Problem 1: List of Complementary Divisors

Develop a program that, given a positive integer, computes the list of its complementary divisor pairs. For a positive integer r, the ordered pair (k, m) is a pair of complementary divisors if $r = km$ and $1 \leq k \leq m$. For example, the complementary divisor pairs of 30 are $(1, 30)$, $(2, 15)$, $(3, 10)$, and $(5, 6)$.

Input: The first line contains a positive integer n indicating how many input values appear on subsequent lines. Each of the next n lines contains a single positive integer.

Output: For each positive integer provided as input, display it, followed by a colon, followed by the list of its complementary divisor pairs, ordered so that the first components are in increasing order. (See below for proper format.)

```
Sample input:
-------------
4
5
36
104
9876


Resultant output:
-----------------
5: (1,5)
36: (1,36) (2,18) (3,12) (4,9) (6,6)
104: (1,104) (2,52) (4,26) (8,13)
9876: (1,9876) (2,4938) (3,3292) (4,2469) (6,1646) (12,823)
```

Problem 2: Line Equations

Develop a program that, given as input two points P and Q on the cartesian plane, outputs the *slope-intercept* equation of the line that passes through P and Q. Recall that the slope-intercept equation of a line has the form

$$y = mx + b$$

so called because m is the slope of the line and b is its *y-intercept* (i.e., the value of y at which the line crosses the y-axis).

In the case that the line described by P and Q has no slope (i.e., is vertical), the program should output an equation of the form $x = c$.

Input: The first line contains a positive integer n indicating how many pairs of points are subsequently given. Each pair of points is described on a single line containing four real numbers: the first two identify P and the last two identify Q. Each point is identified by its x and y coordinates, respectively.

Output: For each pair of points given as input, the slope-intercept equation of the line passing through those two points should appear on a single line of output. (Or, in the case of a vertical line, the equation should be of the form $x = c$.) See the examples below for the proper output format. Notice that if the y-intercept is negative, there should be no plus sign preceding it. Numbers should be accurate to at least three digits.

```
Sample input:
-------------
4
2.0 3.0 -4.0 0.0
-3.7 -2.4 -3.7 6.2
4.5 5.0 0.5 -0.5
5.0 -12.0 -3.0 8.0


Resultant output:
-----------------
y = 0.5x + 2.0
x = -3.7
y = 1.375x - 1.1875
y = -2.5x + 0.5
```

Problem 3: Palindromic Numbers

A number is said to be *palindromic* if, by reversing the order of its digits, we obtain the same number.[1] For example, 484 and 4884 are palindromic, but 3654 is not.

Define r and f to be the functions such that, for positive integer k, $r(k)$ is the number obtained by reversing the order of the digits in k and $f(k) = k + r(k)$.

For example, $r(352) = 253$ and $f(352) = 352 + r(352) = 352 + 253 = 605$.

For positive integer k, define the sequence $\langle k_0, k_1, k_2, \dots \rangle$ as follows: $k_0 = k$ and, for $i > 0$, $k_i = f(k_{i-1})$. That is, the first element in the sequence is k and each element thereafter is obtained by applying f to the previous element.

For example, take $k = 78$. We have

$$\begin{array}{llll} k_0 & = k & = 78 & \\ k_1 & = f(78) & = 78 + 87 & = 165 \\ k_2 & = f(165) & = 165 + 561 & = 726 \\ k_3 & = f(726) & = 726 + 627 & = 1353 \\ k_4 & = f(1353) & = 1353 + 3531 & = 4884 \end{array}$$

We stopped at k_4 because it is palindromic!

Develop a program that, given a positive integer k, computes the first palindromic number in the sequence $\langle k_0, k_1, k_2, \dots \rangle$.

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

Output: For each given k, report its value and the value of the first palindromic number in the sequence $\langle k_0, k_1, k_2, \dots \rangle$, separated by a space, colon, and another space.

You may assume that all given values of k are such that the correct result falls quite comfortably within the range of integers representable using the data type `int` in Java or C/C++.

Sample input and output appear on the next page.

---

[1] A more rigorous definition would say that a number m is palindromic *with respect to a base* b if the base b numeral representing m is a palindrome (i.e., remains the same when its digits are reversed). However, here we will deal exclusively with base 10 (i.e., decimal) numerals and hence we can ignore these details.

```
Sample input:
-------------
8
66
242
35
87
347
249
97
3438


Resultant output:
-----------------
66 : 66
242 : 242
35 : 88
87 : 4884
347 : 1991
249 : 5115
97 : 44044
3438 : 59895
```

University of Scranton
ACM Student Chapter / Computing Sciences Department
26th Annual High School Programming Contest (2016)

Problem 4: Square Transformations

Imagine that you have four unit (i.e., 1×1) squares (labeled **A**, **B**, **C**, and **D**) that have been "glued together" to form a single 2×2 square. There are eight different operations that, when applied to a 2×2 square, preserve the adjacencies between its unit sub-squares while allowing those sub-squares to change positions. They are illustrated below.

A	B
D	C

$\xrightarrow{\wedge}$

A	B
D	C

(Identity (or Full Turn))

A	B
D	C

$\xrightarrow{>}$

D	A
C	B

(Clockwise Quarter Turn)

A	B
D	C

$\xrightarrow{\vee}$

C	D
B	A

(Clockwise Half Turn)

A	B
D	C

$\xrightarrow{<}$

B	C
A	D

(Clockwise Three-Quarter Turn)

A	B
D	C

$\xrightarrow{|}$

B	A
C	D

(Vertical Axis Flip)

A	B
D	C

$\xrightarrow{-}$

D	C
A	B

(Horizontal Axis Flip)

A	B
D	C

$\xrightarrow{/}$

C	B
D	A

(Slash Axis Flip)

A	B
D	C

$\xrightarrow{\backslash}$

A	D
B	C

(Backslash Axis Flip)

If you were to apply two or more operations in sequence, the net effect would be the same as if you had applied a particular one of the eight operations. But which one? Of course, that depends upon the operation sequence.

The illustration below demonstrates that applying a Clockwise Quarter Turn, followed by a Slash Axis Flip, followed by a Horizontal Axis Flip yields the same result as applying a Clockwise Half Turn. Using the symbols by which we have named the various operations, we could express this observation using the equation `>/- = ∨`.

A	B
D	C

$\xrightarrow{>}$

D	A
C	B

$\xrightarrow{/}$

B	A
C	D

$\xrightarrow{-}$

C	D
B	A

Develop a program that, given a character string describing a sequence of operations, indicates which of the eight operations has the same net effect as the given sequence.

Input: The first line contains a positive integer n indicating how many operation sequences will be described on subsequent lines. Each of the next n lines contains a nonempty string describing an operation sequence. Each such string will be composed of characters from the set { `^`, `>`, `v`, `<`, `|`, `-`, `/`, `\` }. Each of these characters represents one of the eight operations, as indicated in the first illustration. (The characters `^` and `v` are used in place of $\wedge$ and $\vee$, respectively, for obvious reasons.)

Output: For each operation sequence given as input, use an equation to identify the single operation to which it is equivalent, following the format exemplified in the sample output shown below.

```
Sample input                Resultant output
------------                ----------------
4
vv                          vv = ^
>/-                         >/- = v
\-<|^v                      \-<|^v = |
/\v<|-<\-v^>>|              /\v<|-<\-v^>>| = /
```

Problem 5: Eight Queens Verification

In the game of *chess*, the *queen* is the most powerful piece, as it can attack along *ranks* (i.e., rows), *files* (i.e., columns), and diagonals of the 8×8 board on which the game is played. (Thus, it combines the abilities of the *rook* and the *bishop*.)

Although it has little or nothing to do with the playing of chess, an interesting problem is that of placing eight queens on the chessboard in such a way that no queen can attack another. In any solution, there will be exactly one queen in each of the eight ranks, exactly one queen in each of the eight files, and at most one queen in each of the 30 diagonals.

One solution (among many) to the problem is illustrated below. Following convention, the chessboard's ranks are numbered 1-8 and its files are labeled **a** through **h**. Squares on which a queen has been placed are marked with a **Q**.

8							Q	
7			Q					
6								Q
5		Q						
4					Q			
3	Q							
2						Q		
1				Q				
	a	b	c	d	e	f	g	h

Figure 1: One Solution to the Eight Queens Problem

This solution is described by the string `dfaebhcg`, which indicates the files in which queens appear, going from ranks 1 through 8.

Develop a program that, given a string of length eight in which every character is one of the lower case letters in the range `a..h`, reports whether or not it describes a valid solution to the Eight Queens Problem.

Input: The first line contains a positive integer n indicating how many proposed solutions will be described on subsequent lines. Each of the next n lines contains a string of length eight, containing only lower case letters between `a` and `h`, inclusive, describing the positions of eight queens on a chessboard. The i-th letter in the string identifies the file in which the queen in rank i appears.

Output: For each input string, echo it and indicate whether or not it describes a valid solution to the Eight Queens Problem, following the format exemplified in the sample output below.

Sample input and output appear on the next page...

```
Sample input:
-------------
7
dfaebhcg
bfhcadge
eahdbgcf
faebhcgd
fdhagbec
dhbacbdg
caehfbdg


Resultant output:
-----------------
dfaebhcg: valid
bfhcadge: valid
eahdbgcf: valid
faebhcgd: valid
fdhagbec: NOT valid
dhbacbdg: NOT valid
caehfbdg: NOT valid
```

Problem 6: Order of a Permutation

A *permutation* of a set is a sequence that contains each member of that set exactly once. For example, $P = \langle c\ a\ b\ d \rangle$ is a permutation of the set $\{a, b, c, d\}$ consisting of the first four lower case letters. If we number the positions in P starting at one, then c is at position 1, a is at position 2, b is at position 3, and d is at position 4. Hence, we can view P as a function that maps 1 to c, 2 to a, and so on. That is, using functional notation, $P(1) = c$, $P(2) = a$, $P(3) = b$, and $P(4) = d$. To emphasize this point of view, we can express P like this:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c & a & b & d \end{pmatrix}$$

For this problem we are concerned only with permutations of the set $\mathbb{Z}_m^+ = \{1, 2, \ldots, m\}$, where m is some positive integer. (Thus, henceforth any use of the term "permutation" is to be understood to mean one of these.) As an example, here is a permutation of the set $\mathbb{Z}_7^+$:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 5 & 7 & 1 & 2 & 6 & 3 & 4 \end{pmatrix}$$

We can generate new permutations by *composing* a permutation with itself any number of times. Specifically, if σ is a permutation of $\mathbb{Z}_m^+$ (where m is any positive integer) and $k \geq 0$, we define $\sigma^{[k]}$ to be the permutation such that for all $j \in \mathbb{Z}_m^+$,

$$\sigma^{[k]}(j) \ = \ \sigma(\sigma(\ldots(\sigma(j))\ldots))$$

where σ appears on the right-hand side exactly k times. For example, taking σ to be the permutation depicted above,

$$\sigma^{[3]}(6) \ = \ \sigma(\sigma(\sigma(6))) \ = \ \sigma(\sigma(3)) \ = \ \sigma(1) \ = \ 5$$

It turns out that, for any permutation of $\mathbb{Z}_m^+$, repeated composition will eventually produce the *identity* permutation, which maps each of $1, 2, \ldots, m$ to itself. The minimum number of compositions necessary to achieve this is referred to as the *order* of the permutation. Our example permutation σ has order 12, as $\sigma^{[12]}$ is the identity permutation but none among $\sigma^{[1]}$, $\sigma^{[2]}$, ..., $\sigma^{[11]}$ is the identity permutation.

Develop a program that, given as input a positive integer m and a permutation of the set $\mathbb{Z}_m^+$, reports the order of that permutation.

Hint: The order of a permutation is a function of the lengths of its cycles. Our example permutation has two cycles, one of length four (involving $1, 3, 5, 6$) and one of length three (involving $2, 4, 7$).

Input: The first line contains a positive integer n indicating how many permutations are described on the succeeding n lines. Each instance of the problem is described on two lines, the first of which contains a positive integer m and the second of which contains a permutation σ of the set $\{1, 2, \ldots, m\}$ described by the sequence of numbers $\sigma(1)$, $\sigma(2)$, ..., $\sigma(m)$.

Output: For each permutation given as input, the program is to display the permutation and its order on one line. See the sample output below for proper formatting.

```
Sample input:
-------------
5
7
5 7 1 2 6 3 4
11
4 10 7 11 2 1 5 8 3 6 9
11
4 10 3 11 8 7 5 6 9 2 1
11
5 2 1 11 8 4 6 3 10 7 9
8
5 8 4 6 1 7 3 2

Resultant output:
-----------------
Permutation <5 7 1 2 6 3 4> has order 12
Permutation <4 10 7 11 2 1 5 8 3 6 9> has order 10
Permutation <4 10 3 11 8 7 5 6 9 2 1> has order 12
Permutation <5 2 1 11 8 4 6 3 10 7 9> has order 12
Permutation <5 8 4 6 1 7 3 2> has order 4
```

Problem 7: Polynomial Division

Develop a program that computes the quotient and remainder resulting from dividing a polynomial of the form $x + a$, where a is an integer, into a polynomial with integer coefficients. To illustrate, here are two worked-out examples:

```
      2x^2 + 2x +  7
    ----------------------
x-3 ) 2x^3 - 4x^2 + 1x -  3
      2x^3 - 6x^2
      --------------------
             2x^2 + 1x -  3
             2x^2 - 6x
             -------------
                    7x -  3
                    7x - 21
                    -------
                         18
```

```
        4x^3 -  6x^2 + 12x -  29
      --------------------------------
x+2 ) 4x^4 + 2x^3 +  0x^2 -  5x +  2
        4x^4 + 8x^3
        ------------------------------
              -6x^3 +  0x^2 -  5x +  2
              -6x^3 - 12x^2
              ------------------------
                      12x^2 -  5x +  2
                      12x^2 + 24x
                      ---------------
                             -29x +  2
                             -29x - 58
                             ---------
                                    60
```

In the example on the left, $x - 3$ is the divisor and $2x^3 - 4x^2 + x - 3$ is the dividend. The resulting quotient and remainder are $2x^2 + 2x + 7$ and 18, respectively.

Input: The first line contains a positive integer n indicating how many division problems are described on succeeding lines. Each division problem is described on two lines, the first of which contains the constant a in the divisor $x + a$ and the second of which contains the degree d of the dividend followed by its $d+1$ coefficients, in order from the most to the least significant term.

Output: For each division problem given as input, display it, together with the quotient and remainder, in the format exemplified below. If you were to display the polynomials without using unary minus signs and/or omitting 1 coefficients and/or omitting altogether terms with zero coefficients, that would be preferred, but not required. For example, $2x^4 - 5x^2 + x - 3$ is a better way to write $2x^3 + -5x^2 + 1x + -3$. (Note that the first example output line below is in an acceptable format and the second example output line is in the preferred but not required format.)

```
Sample input:              Explanation:
------------               ------------
2                          Two division problems will be described
-3                         divisor is x - 3
3 2 -4 1 -3                dividend has degree 3 with coeffients 2,-4,1,-3
2                          divisor is x + 2
4 4 2 0 -5 2               dividend has degree 4 with coefficients 4,2,0,-5,2


Resultant output:
-----------------
(2x^3 + -4x^2 + 1x + -3) / (x + -3) = 2x^2 + 2x + 7 remainder 18
(4x^4 + 2x^3 - 5x + 2) / (x + 2) = 4x^3 - 6x^2 + 12x - 29 remainder 60
```

Problem 8: NFA String Acceptance

Depicted in the figure below is a *nondeterministic finite automaton* (NFA) that we will refer to as M. Each circle represents a *state* and each arrow labeled by a symbol represents a *transition* from one state to another (or itself) associated with that symbol. The unlabeled arrow points to the *initial state*. Double circles correspond to *final states*. By convention, we name the states in an NFA q_0, q_1, ..., q_{m-1}, where m is the number of states and q_0 is the initial state. The set of symbols appearing as labels on transitions is called the *alphabet* of the NFA.

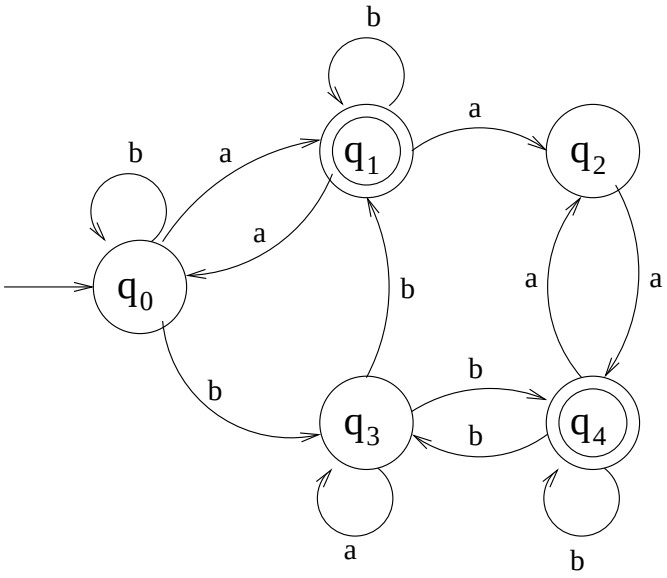

Figure 2: Nondeterministic Finite State Machine M

Consider the string *baab*. In M there are three *paths* beginning at the initial state that "spell out" that string:

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_0 \xrightarrow{b} q_0$$
$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_0 \xrightarrow{b} q_3$$
$$q_0 \xrightarrow{b} q_3 \xrightarrow{a} q_3 \xrightarrow{a} q_3 \xrightarrow{b} q_4$$

Due to the fact that at least one of these paths (namely, the last one) ends in a final state, M is said to *accept baab*.

If, on the other hand, all paths spelling out a particular string end in non-final states, that string is said to be *rejected* by M. An example is the string *baa*, as the three paths that spell it out end in the non-final states q_0, q_2, and q_3, respectively.

Notice that it is possible for there to be *no* paths that spell out a particular string, in which case it is certainly rejected. An example of this is aab (or any string of which aab is a prefix). (The only path spelling out aa ends in q_2, but there is no outgoing transition from there labeled b; hence, even if q_2 were a final state, any string having aab as a prefix is rejected.)

Develop a program that, given an NFA with alphabet $\{a, b\}$ and some input strings composed of a's and b's, reports, for each input string, whether or not it is accepted by the NFA.

Input: The first line contains the number $m > 0$ of states in the NFA to be tested, followed by the number $k > 0$ of final states, followed by the number $t > 0$ of transitions. The second line contains k integers in the range $0..m-1$ identifying the NFA's final states. (The initial state is not explicitly identified because it is to be understood that it is state 0.) Each of the following t lines describes a single transition, which is given by two state identifiers (integers in the range $0..m-1$) p and q separated by the label (either a or b) on a transition that goes from state p to state q. The transitions need not be listed in any particular order. (The sample input below describes M.)

On the following line is a positive integer r indicating how many input strings are to be processed. Each of the following r lines contains one such string.

Output: For each string given as input, generate one line of output that classifies that string as being either accepted or rejected by the NFA.

```
Sample Input:
-------------
5 2 13
1 4
0 b 0
0 b 3
4 b 4
3 a 3
0 a 1
3 b 1
1 a 0
1 b 1
3 b 4
1 a 2
2 a 4
4 a 2
4 b 3
6
baab
aabab
bbbb
bbbababaa
aaaaaa
abbaaabaa
```

```
Resultant Output:
-----------------
baab is accepted
aabab is accepted
bbbb is accepted
bbbababaa is accepted
aaaaaa is rejected
abbaaabaa is rejected
```