The Twentieth Annual International Collegiate Programming Contest

Problem A: Division of N-Digit, M-Based Integers

Input File: jtd1.dat

Suppose A and B are two m-based integers with N digits, where $1 \leqq N \leqq 80$ and $2 \leqq m \leqq 20$. For examples, $(567908)_{10}$ is a 6-digit 10-based integer and $(AF5012B)_{16}$ is a 7-digit 16-based integer. Please write a program to calculate the quotient Q of A divided by B, where Q is an m-based real number with F digits of fractional part and $0 \leqq F \leqq 9$.

Input

The 1st line is the base 'm' (represented in a decimal integer).

The 2nd line and the 3rd line are the dividend 'A' and the divisor 'B' respectively (represented in m-base integers).

The 4th line is 'F' (represented in a decimal integer).

Output

The quotient Q. It is an m-based real number with the precision of F digits of fractional part. The fractional part after the F-th fractional digit will be truncated.

Note:

1. For an m-base integer when m<10, the digits are represented by '0', '1', '2', ..., and m-1. For an m-base integer when $m \leqq 10$, the digits should be represented by '0', '1', '2', ..., '9', 'A', 'B', ..., and so on.
2. The program should accept multiple sets of data.

Sample Input

```
16
29FA5
7
3
8
7654321
12345
5
18
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
11111111111111111111111
6
```

Sample Output

```
5FF3.000
600.04020
H000000000000000000000.00GHH
```

The Twentieth Annual International Collegiate Programming Contest

Problem B: River Crossing

Input File: jtd2.dat

M *wolves* and N *dogs*, where **N** $\geqq$ **M**, need to cross a river from its west bank to the east bank using a boat which can hold **<u>at most two</u>** individuals. Safety constraint: the number of *dogs*, unless it is zero, is never less than that of *wolves* on either bank or on the boat. General constraint: the boat needs to carry **<u>at least one</u>** individual on its way to the east bank or back to the west bank. Is it possible to cross the river without the *dogs* being eaten? If yes, how can we move them? Give one legal moving sequence.

This problem can be modeled by using simply the information of the occupants on the west bank, and the position of the boat. Let us use **(X,Y,P)-triples** to denote different states in the process of moving, where **X Y** and **P** are the numbers of *wolves* and *dogs* on the west bank and the position (**west W or east E**) of the boat at beginning or after a legal move, respectively. For example, the inital state is **(M,N,W)** and the final state is **(0,0,E)**. A safety state **(m,n,P)** for this problem will be **n** $\geqq$ **m**, unless n = 0, and **(N - n)** $\geqq$ **(M - m)**, unless N - n = 0 (i.e., the number of *dogs* on either the west bank or the east bank is never less than that of the *wolves* at any time).

Write a program given three pairs of the beginning numbers of *wolves* and *dogs* (stored in the file **river.dat**) will output three results, separately. Each result is either a legal moving sequence, <u>an ordered transition from the initial safe state **(M,N,W)** to the final state **(0,0,E)**</u>, if there exists a solution, or the message '**Impossible**.' if no any solution exists. Meanwhile, a circular (or useless) moving process, i.e., from a state to a state itself after zero or several moves (transitions), is not allowed to be included in a legal moving sequence.

For example, let M = 2 and N = 2. One legal moving sequence is
(2,2,W) $\leqq$ (0,2,E) $\leqq$ (1,2,W) $\leqq$ (1,0,E) $\leqq$ (2,0,W) $\leqq$ (0,0,E).

Input
The input data format in the input file looks like this:

w1 d1

w2 d2

w3 d3

, where w1 d1, w2 d2, and w3 d3 are the beginning numbers (integers) of *wolves* and *dogs* in three different lines. There is one space between each of the w1 d1, w2 d2 and w3 d3 pairs.

Output
The format of output for a legal moving sequence is a set of an **ordered** states listed line by line. For example, the output of the aforementioned example is

(2,2,W)

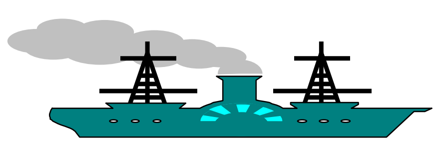
(0,2,E)

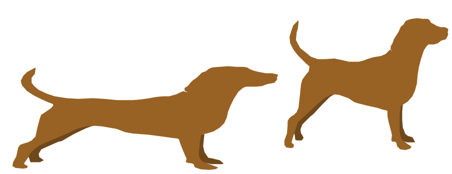
(1,2,W)

(1,0,E)

(2,0,W)

(0,0,E)





One or more blank lines are required for separating each of the output results.

The Twentieth Annual International Collegiate Programming Contest

Problem C: Shortest Paths

Input File: jtd3.dat

Given a directed graph G=(V,E), and a source vertex $v_0 \in V$, please write a program to output the lengths of the shortest paths between v_0 and all other vertices in the graph G. We assume the edges in digraph G can have negative costs, provided that the sum of the costs around any cycle in the digraph is positive. The vertices are labeled from 1 to n, if G has n vertices. The data formats of the input and the output of this problem are arranged as follows.

Input

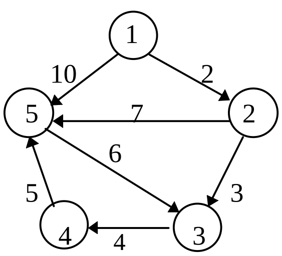
Your program must accept the following input data from input file **jtd3.dat**
(1) A positive integer n, it stands for the total number vertices of G.
(2) A source vertex v_0 ($v_0 \in V$, i.e., it could be any vertex in G.)
(3) The cost matrix W of G. It is arranged by row-wise order as follows.

the first row of cost matrix W (enter ↵)
the second row of cost matrix W (enter ↵)
: .
the last row of cost matrix W (enter ↵)

Output
The lengths of the shortest paths from v_0 to all other vertices v_i, the format is (v_0 -> v_i) = ?

Notes:
(1) In the input, we arrange the input data $W(v_i, v_j)$ = '-', if (v_i, v_j) is not an edge in G, $v_i \neq v_j$, and $W(v_i, v_j) = 0$, if $v_i = v_j$.
(2) Your program must be able to accept the multiple set of input data.

Example:
Assume we have the following 5-vertex digraph G and its cost matrix W.



$$W = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 2 & - & - & 10 \\ - & 0 & 3 & - & 7 \\ - & - & 0 & 4 & - \\ - & - & - & 0 & 5 \\ - & - & 6 & - & 0 \end{bmatrix} \end{array}$$

, where $W(v_i, v_j)$='-' means there is no edge between v_i and v_j in G.

If we let v_0 = 1, then your input data must be

5
1
0 2 - - 10
- 0 3 - 7
- - 0 4 -
- - - 0 5
- - 6 - 0

and the output would be

(1 -> 2) = 2
(1 -> 3) = 5
(1 -> 4) = 9
(1 -> 5) = 9

The Twentieth Annual International Collegiate Programming Contest

Problem D: List Search

Input File: jtd4.dat

Following the conventions of LISP assume a list contains two fields HEAD and TAIL. If a list *A* = ((a(bc))), then HEAD(*A*) = (a(bc)), TAIL(*A*) = NIL, HEAD(HEAD(*A*)) = a, TAIL(HEAD(*A*))=((bc)). Write a program to read in a source list and a target list, then print out a sequence of proper HEAD, TAIL operations for extracting the target list from the source list.

For instance, the following **input** demands to find (a) from the list (bc((a))).

(bc((a)))
(a)

The correct **output** for the above problem is:

TAIL
TAIL
HEAD
HEAD

Assumptions:
1) All symbols are single character.
2) There are no duplicate symbols in the source list and the target list.

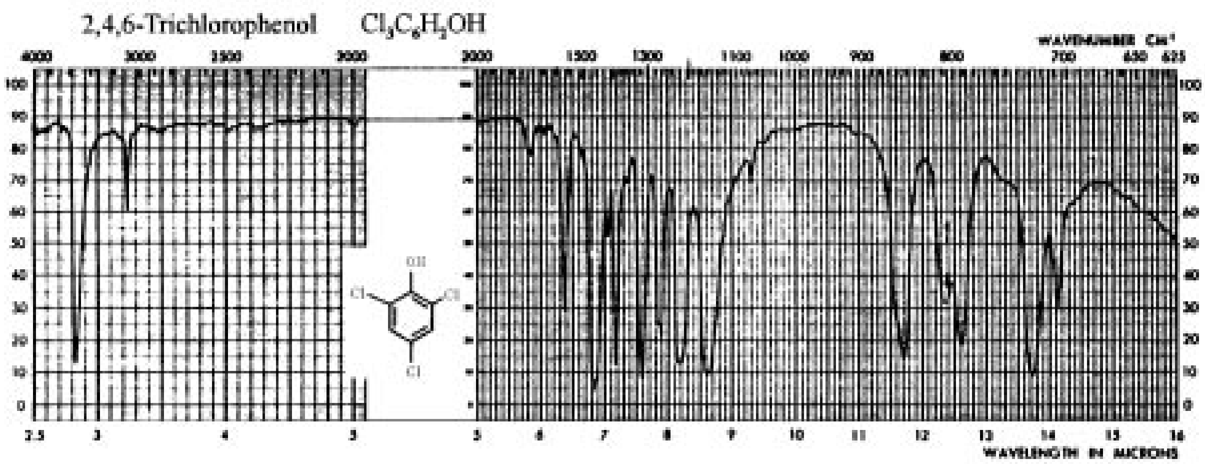
Note:
The program should accept multiple sets of data.

The Twentieth Annual International Collegiate Programming Contest

Problem E: Type Matching

Input File: jtd5.dat

Infrared(IR) spectrum, frequently used in identifying organic residues in environment, is characterized by a set of camel back shape peaks, and each peak is represented by its adsorption frequency (or wavenumber, cm^{-1}) and intensity (i.e. strong, medium, and weak). An IR adsorption spectrum of 2,4,6-trichlorophenol, a highly toxic chemical regulated by EPA, is shown below. It is unlikely that two different compounds have identical IR spectrum, and it is the basis we use to identify unknown compounds by type matching or "pattern recognition".



This type matching is best done by an elimination process: comparing the IR adsorption of the unknown compound with those from known compounds and eliminating the wrong match, then the remaining ones are possible candidates. This process is continued until a perfect match is met

You are required to write a program to do the matching process. IR spectra, including peak position (cm^{-1}) and peak intensity (s, m, or w), of twenty-two organic compounds are given in the table next page. You will use them to identify unknown compounds. It is not necessary to have a perfect match, you are only required to recognize all the possible (candidates).

Hint: Two steps in identifying organic compounds.

1. Check the appearance of all the characteristic peaks.
2. Check the relative intensity of every characteristic peaks.

Any mismatch of characteristic peak positions or intensities between IR spectra should correspond to different compounds.

Input: The input data of each test comes from a file, that consists of total number of adsorption peaks, peak position and peak intensity. For example, if the total number of peaks is N, the input file is as follows:

```
N          //total number of peaks
p1  i1     //position and intensity of first adsorption peak
p2  i2     //position and intensity of second adsorption peak
...  ...                ........
pN  iN     //position and intensity of Nth adsorption peak
```

Output: print out the type of organic compound.

Example:

Peaks:880(w), 1100(w), 1345(w), 1396(w), 1437(m), 1462(m), 2250(m), 2890(s), 2950(s), 2975(s)

Identification result: Alkanes; Nitriles

Its input and output is as follows:

Input:	**Output:**
10	Alkanes; Nitriles (or Nitriles; Alkanes)
880 w	
1100 w	
1345 w	
1396 w	
1437 m	
1462 m	
2250 m	
2890 s	
2950 s	
2975 s	

Note: The program should accept multiple sets of data.

Type of organic compounds	Peak position (cm^{-1})	Peak intensity	Type of organic compounds	Peak position (cm^{-1})	Peak intensity
Alkanes	2850~3000	s	Benzene	1460~1530	m
Alkynes	3300	s		3001~3100	m or s
	2100~2250	m or w	Monosubstituted benzenes	675~710	s
Aldehydes	1000~1300	m		730~765	s
	1650~1740	s		1460~1530	m
	2700~2800	w		1550~1650	m
	2801~2900	w		3001~3100	m or s
Ketones	1000~1300	s	o-Disubstituted benzenes	730~755	s
	1650~1740	s		1460~1530	m
Carboxlic acids	1000~1300	s		1550~1650	m
	1700~1725	s		3001~3100	m or s
	2400~3400	m	m-Disubstituted benzenes	660~710	s
Anhydrides	1050~1175	s		750~810	s
	1710~1780	s		855~900	m
	1781~1845	s		1460~1530	m
Cyclic anhydrides	1210~1310	s		1550~1650	m
	1730~1805	s		3001~3100	m or s
	1806~1875	s	p-Disubstituted benzenes	800~855	s
Alcohols	1000~1300	s		1460~1530	m
	3101~3500	s		1550~1650	m
Nitriles	2240~2260	m		3001~3100	m or s
Monosubstituted alkenes	525~700	s	1,2,4-Trisubstituted benzenes	780~840	s
	900~1000	s		865~895	m
	1600~1680	m or w		1460~1530	m
	3001~3100	m		1550~1650	m
cis-1,2-Disubstituted alkenes	700~750	s		3001~3100	m or s
	1600~1680	m or w	1,2,3-Trisubstituted benzenes	680~720	m
	3001~3100	m		745~785	s
trans-1,2-Disubstituted alkenes	950~1000	s		1460~1530	m
	1600~1680	m or w		1550~1650	m
	3001~3100	m		3001~3100	m or s
1,1-Disubstituted alkenes	875~925	s	1,3,5-Trisubstituted benzenes	675~725	m
	1600~1680	m or w		830~910	s
	3001~3100	m		1460~1530	m
Trisubstituted alkenes	800~825	m		1550~1650	m
	1600~1680	m or w		3001~3100	m or s
	3001~3100	m			

Test: Please identify following peaks

1.Peaks:667(s), 1030(m), 1475(m), 1800(m), 1950(m), 3010(s), 3090(m), 3100(m)
Identification result: Benzene
2.Peaks:685(s), 800(w), 870(w), 965(s), 1090(m), 1405(s), 1600(w), 1920(w), 2245(m), 2255(w), 3010(m), 3050(m)
Identification result: Nitriles; Monosubstituted alkenes
3.Peaks:705(m), 780(w), 862(s), 885(m), 1080(m), 1170(w), 1320(m), 1380(m), 1460(m), 1600(m), 1750(w), 2890(s), 2910(s), 2990(s), 3010(m)
Identification result: Alkanes; Benzene; 1,3,5-Trisubstituted benzenes
4.Peaks:1380(m), 1468(m), 2852(s), 2930(s), 2960(s)
Identification result: Alkanes
5.Peaks:680(s), 760(s), 955(m), 1020(m), 1080(m), 1178(m),1264(s), 1300(m), 1360(m), 1460(m), 1580(m), 1600(m), 1680(s), 3005(m), 3060(m)
Identification result: Ketones; Benzene; Monosubstituded benzenes

The Twentieth Annual International Collegiate Programming Contest

Problem F: Roman Numbers

Input File: jtd6.dat

Write a program to convert positive integers into Roman numbers. Assume that the numbers to be converted is less than 5000. The rule for constructing a Roman number is assumed to be as follows. In Roman number system, **i** is the symbol for 1, **v** for 5, **x** for 10, **l** for 50, **c** for 100, **d** for 500 and **m** for 1000. Symbols with larger values usually appear before symbols with smaller values. The value of a Roman number is, in general, the sum of the values of the symbols. For example, **ii** is 2, **viii** is 8. However, if a symbol with smaller value appears before a symbol with larger value, the value of these two symbols is the difference of the two values. For example, **iv** is 4, **ix** is 9, and **lix** is 59. Note that no four consecutive symbols in the Roman number can be the same. For example, **iv**, but not **iiii**, is the Roman number 4. The Roman numbers constructed in this way may not be unique. For example, both **mcmxc** and **mxm** are valid for 1990. Although the roman number generated by your program need not be the shotrtest one, never use **vv** for 10, **ll** for 100, **dd** for 1000, or **vvv** for 15, etc.

Input

The input data is stored in a file. Each record of the input file contains one positive integer x. You may assume that x is less than 5000.

Output

For each number of the input file, print the number in decimal and its Roman form.

Sample Input

```
3
8
172
4
1990
5
```

Sample Output

```
3      iii
8      viii
172    clxxii
4      iv
1990   mcmxc
5      v
```