ACM South American Regional
Collegiate Programming Contest

Contest Session

November 14, 1998

São Paulo – Brazil
Antofagasta – Chile
Caracas – Venezuela

acm International Collegiate Programming Contest '98-99
IBM event sponsor
HOME | INFORMATION | PEOPLE | PAST
REGIONALS | MAP | WORLD FINALS | KEEP ME INFORMED

PROBLEM 1: ALGOC - A Language to Generate Only Constants

File Names

Source File: algoc.pas or algoc.c or algoc.cpp
Input File: algoc.in
Output File: algoc.out

Statement of the Problem

An obscure Computer Science professor wants to become famous developing a new programming language, ALGOC – A Language to Generate Only Constants. This language is very simple, and has only four constructs:

- PLUSONE - creates the constant 1 (positive).
- MINUSONE - creates the constant -1 (negative).
- INC - adds one to the constant being generated.
- DUP - multiplies the constant being generate by two.

A program in this language is a sequence of these constructs, one in each line, executed sequentially. The professor wants to keep the programs written in this language simple, small and fast. To achieve his goal, he adds the following constraints:

- Every program must begin either with PLUSONE or with MINUSONE.
- A given constant C must be generated with the smallest number of instructions possible.
- If a constant C can be generated by different programs (all with equal number of instructions), then the fastest program should be used. For this purpose, suppose that the instruction DUP is executed in T nanoseconds and the instruction INC in 2T nanoseconds.

You were hired by the professor to write several sample programs, so that he can use them in various conferences to demonstrate his powerful new language. The professor will give you a few constants and your task is to write programs to generate those constants, obeying the constraints above.

Input Format

The input file may contain several instances of the problem. Each instance of the problem is just one line containing the numeric constant to be generated. All numbers are **nonzero** integers between -32768 and 32767. A line containing the integer zero terminates the input file.

Output Format

For each instance of the problem, your program should print one line saying `Constant XXX`, where XXX is the constant for that instance, followed by the most efficient program to generate that constant, one instruction per line.

Each output of an instance is terminated with a blank line.

Sample Input

```
3
-5
1
7
0
```

Sample Output

```
Constant 3
PLUSONE
DUP
INC

Constant -5
MINUSONE
DUP
DUP
INC
DUP
INC

Constant 1
PLUSONE

Constant 7
PLUSONE
DUP
INC
DUP
INC
```

PROBLEM 2: Inflated Numbers

File Names

Source File: inflation.pas or inflation.c or inflation.cpp
Input File: inflation.in
Output File: inflation.out

Statement of the Problem

In a land ruled by crazy economists there is the greatest inflation in human history. Trying to control it, the government has issued the following laws:

1. All quantities should be represented by integer numbers in base 10 (thereis no fractional cents). Numbers in scientific notation are copyrighted by the INA (International Number Association) and thus should not be used, to avoid currency evasion.
2. The arithmetical operations should be able to handle any arbitrarily long signed integer.
3. Multiplication and division are operations used only by extremists and should not be used; only addition and subtraction are allowed.

Devising a big consumer market, a multinational wants to produce pocket calculators to be used in this country. You are hired to write a program to perform addition and subtraction over big integers (luckily to you, another team was hired to develop a display that can show these numbers).

Input Format

The input file may contain several instances of the problem.

1. Each instance is a sequence of characters, representing an operation in the following format:

   `integer [one or more spaces] operator [one or more spaces] integer`

   where `integer` is a sequence of digits ended by the symbol $.
2. Negative numbers are immediately preceded by a minus symbol (-). Operators are one of $+, -$. As the integers can be arbitrary long, line changes should be ignored in the input.

Every instance begins in a new line.
The input file is terminated by a line containing only the symbol #.

Output Format

For each instance of the problem, your program should print the number resulting from the operation, with sixty digits per line, except, perhaps, the last one.
The output of each instance is terminated with a blank line.

Sample Input

```
123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789$ - 1234567
89$
-2$ + 5000000000000000000000000000000000000$
#
```

Sample Output

```
123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890000000000

4999999999999999999999999999999999998
```

PROBLEM 3: Help the Museum

File Names

Source File: museum.pas or museum.c or museum.cpp
Input File: museum.in
Output File: museum.out

Statement of the Problem

The National Association of Museum Curators came to you with an interesting problem. The President of the country, in order to improve his public image, has decided to visit the various Art Museums in the country, to convey the impression that he is a refined man. Being a very busy person, however, and knowing nothing about art, he imposed two restrictions for the visits:

1. In each museum, he wants to see the works of one and only one artist, so that he can easily prepare himself for the visit and pose as an art connoisseur. However he does not necessarily have to see all of the works of that artist.
2. He does not want to waste time, and demands to walk through the exposition following the shortest possible path.

The curators are willing to follow the President's demands, but they do not want to redistribute the masterpieces in the exposition only to obtain a straight path. Their only concession is to exchange temporarily the place of two masterpieces, if this helps to obtain a shorter path.

You should write a program that receives as input the layout of an exposition and finds the shortest path, according to the above constraints. To make your task easier, the curators have already defined a standard layout. Figure 1 shows one such layout.

10	B	B	B	B	B	B	F	F	F	F	F
9	A	A	A	A	A	B	D	C	C	F	F
8	A	F	F	F	A	B	A	A	C	F	C
7	B	F	E	F	A	B	B	B	B	B	D
6	F	F	D	E	A	B	A	A	A	B	A
5	E	E	D	E	E	E	E	E	A	B	B
4	D	D	D	E	E	E	E	E	A	A	B
3	D	C	C	F	F	F	C	C	A	B	A
2	D	C	C	F	F	F	C	C	A	A	A
1	C	C	C	C	C	C	C	C	C	C	C
Y/X	1	2	3	4	5	6	7	8	9	10	11

Figure 1: Layout of the Museum

The President's walk begins always at the left wall ($X = 1$, any Y) and ends at the right wall ($X = X_{\mathrm{max}}$, any Y). The walk can be done horizontally or vertically; diagonal movements are not allowed. The works of a given artist are all labeled with the same capital letter (A, B, C, etc). From Figure 1, several cases can be illustrated:

1. If the president wants to see the works of artist A, there is not a path from left to right. Such a path can be obtained if the work by artist B at (6,6) is exchanged with one by artist A from (1,8), (7,8), (8,8), (10,6), (11,6) or (11,3).
2. If the president wants to see the works of artist B, there is already a path, beginning at (1,10) and ending at (11,5). A shorter path can be obtained, if the work of D at (11,7) is exchanged with one work by artist B (for example, from (10,6)).

3. If the president wants to see the works of artist C, there is already a straight path from (1,1) to (1,11), and a shorter path cannot be obtained.
4. If the president wants to see the works of D, E or F, there is no possibility to obtain a path from left to right.

Input Format

The input file may contain several instances of the problem. Each instance has the following format (all numbers are positive integers):

1. The first line contains the integers $X_{\max}$ and $Y_{\max}$, the layout dimensions. You may assume that $1 \leq X_{\max}, Y_{\max} \leq 100$.
2. The second line contains the artist (upper-case) letter that will have his/her works visited.
3. $Y_{\max}$ lines, each with $X_{\max}$ letters (without spaces between them). The first input line corresponds to index $Y_{\max}$, the second line to index $Y_{\max} - 1$, and so on, until the last line, that corresponds to index 1.

A line containing the integers `0 0` terminates the input file. Numbers are separated by an arbitrary number of spaces.

Output Format

For each instance of the problem, your program should first print one line with the message "Exchange (x,y) and (u,v)" if an exchange will occur, or "No exchange", otherwise.

Following that, the path is output, one coordinate per line. If a path doesn't exist, "No path" should be printed.

The output of each instance is terminated with a blank line.

Sample Input

For the example in Figure 1 suppose the following input:

```
11 10
A
BBBBBBFFFFF
AAAAABDCCFF
AFFFABAACFC
BFEFABBBBBD
FFDEABAAABA
EEDEEEEEABB
DDDEEEEEAAB
DCCFFFCCABA
DCCFFFCCAAA
CCCCCCCCCCC
11 10
C
BBBBBBFFFFF
AAAAABDCCFF
AFFFABAACFC
BFEFABBBBBD
FFDEABAAABA
```

```
EEDEEEEEABB
DDDEEEEEAAB
DCCFFFCCABA
DCCFFFCCAAA
CCCCCCCCCCC
0 0
```

Sample Output

The corresponding output is:

```
Exchange (6,6) and (1,8)
1 9
2 9
3 9
4 9
5 9
5 8
5 7
5 6
6 6
7 6
8 6
9 6
9 5
9 4
9 3
9 2
10 2
11 2

No exchange
1 1
2 1
3 1
4 1
5 1
6 1
7 1
8 1
9 1
10 1
11 1
```

PROBLEM 4: Visible Squares

File Names

Source File: squares.pas or squares.c or squares.cpp
Input File: squares.in
Output File: squares.out

Statement of the Problem

You are given n squares in the coordinate plane whose sides are parallel to the coordinate axes. All the corners have integer coordinates and the squares do not touch or overlap.

A square is *visible* from a point O, if there are two distinct points A and B on one of its sides, such that the interior of the triangle OAB has no common points with any of the remaining squares.

You are required to count the number of squares visible from the origin point $(0, 0)$.

Input Format

The input file may contain several instances of the problem. Each instance is described as follows:

1. The first line contains the integer n, $1 \leq n \leq 1000$, representing the number of squares.
2. Each of the following n lines describes a square: it contains a triple of integers $x\ y\ l$, $1 \leq x, y, l \leq 10000$, where coordinate (x, y) is the lower left corner (the corner with the least x and y coordinates) and l is the side length of that square.

The input file is terminated by a line containing only the integer 0 (zero). All numbers are separated by an arbitrary number of blank spaces.

Output Format

For each instance of the problem, your program should print just one line, containing the number of squares visible from the origin.

The output of each instance is terminated with a blank line.

Sample Input

```
3
2 6 3
1 4 1
3 4 1
4
1 2 1
3 1 1
2 4 2
3 7 1
0
```

Sample Output

```
3

2
```

PROBLEM 5: Cellular Automata

File Names

Source File: cell.pas or cell.c or cell.cpp
Input File: cell.in
Output File: cell.out

Statement of the Problem

A *cellular automaton* is a tuple (Q, f), where Q is a finite set of *states* and $f : Q^3 \longrightarrow Q$.

A *configuration of length n* is a sequence of the form $c_1 \cdots c_n$ such that every c_i is an element of Q. We say that a configuration $d_1 \cdots d_n$ is a *successor* of $c_1 \cdots c_n$ if each d_i was constructed by means of the rule:

$$d_i = \begin{cases} f(c_n, c_1, c_2) & \text{if } i = 1; \\ f(c_{i-1}, c_i, c_{i+1}) & \text{if } 1 < i < n; \\ f(c_{n-1}, c_n, c_1) & \text{if } i = n. \end{cases}$$

In this problem you have to write a program to find the function f for a cellular automaton, given a sequence of configurations $S_1, S_2, \ldots S_k$, where S_{i+1} is a successor configuration of S_i.

Input Format

The input file may contain several instances of the problem. For each instance,

1. the first line contains two integers n and m, indicating, respectively, the number of states and the length of the configurations of the automaton. The states of the automaton are then taken to be $1, 2, \ldots, n$. The values of n and m are such that $1 \leq n \leq 10$ and $1 \leq m \leq 30$;

2. the following lines contain a sequence of configurations, each a successor of the previous one (except for the first one, of course), in the format:

   $$c_1 \ c_2 \ \ldots \ c_m$$

   where $1 \leq c_i \leq n$.

Each instance is terminated by a line containing m zeroes.

All numbers are separated by one or more blank spaces.

Output Format

For each instance of the problem, the output consists of the description of the function f, written as a list of elements, each of the form $q_1 \ q_2 \ q_3 : q$, indicating that $f(q_1, q_2, q_3) = q$, or a message `incomplete information`, saying that it is not possible to determine f from the information given in the file. **Your output must list the function f by increasing lexicographical order of the triplets q_1, q_2, q_3.**

Each output of an instance should be terminated with a blank line.

Sample Input

```
2 7
1 1 2 2 2 1 1
1 2 1 1 1 2 1
2 2 2 1 2 2 2
1 1 1 1 1 1 1
0 0 0 0 0 0 0
```

```
2 7
1 1 2 2  2  1   1
1 2 1 1 1 2 1
0  0 0 0 0 0 0
```

Sample Output

```
1 1 1 : 1
1 1 2 : 2
1 2 1 : 2
2 1 1 : 2
1 2 2 : 1
2 1 2 : 1
2 2 1 : 1
2 2 2 : 1

Incomplete information
```

PROBLEM 6: Contracting Integers

File Names

Source File: contract.pas or contract.c or contract.cpp
Input File: contract.in
Output File: contract.out

Statement of the Problem

You are given a sequence of n positive integers $a = [a_, a_2, \ldots, a_n]$, on which you can perform contraction operations. One *contraction* operation consists of replacing adjacent elements a_i and a_{i+1} by their difference $a_i - a_{i+1}$. More formally, let $\mathrm{con}(a, i)$ denote the $(n-1)$-element sequence obtained from sequence a by the contraction described above, involving elements a_i and a_{i+1}. That is:

$$\mathrm{con}(a, i) = [a_1, \ldots, a_{i-1}, a_i - a_{i+1}, a_{i+2}, \ldots, a_n]$$

Applying $n - 1$ contractions to any given sequence of n integers obviously yields a single integer. For example, applying contractions 2, 3, 2 and 1, in that order, to the sequence $[12, 10, 4, 3, 5]$ yields the integer 4, since:

$$\begin{aligned}
\mathrm{con}([12, 10, 4, 3, 5], 2) &= [12, 6, 3, 5] \\
\mathrm{con}([12, 6, 3, 5], 3) &= [12, 6, -2] \\
\mathrm{con}([12, 6, -2], 2) &= [12, 8] \\
\mathrm{con}([12, 8], 1) &= [4]
\end{aligned}$$

Given a sequence $a_1, a_2, \ldots, a_n$ and a target number T, the problem is to find a sequence of $n - 1$ contractions which, when applied to the original sequence, yields T. We assume that there is always one such sequence of contractions yielding T.

Input Format

The input file may contain several instances of the problem. Each instance is given as follows:

1. The first line contains two integers, separated by an arbitrary number of blank spaces: the integer $n, 1 \le n \le 100$ (the number of integers in the original sequence), and the target integer $T, -10000 \le T \le 10000$.
2. The following n lines contain the starting sequence, i.e., a sequence of n integers a_i, one per line, where $1 \le a_i \le 100$.

The input file is terminated by a line containing the integers `0 0`.

Output Format

For each instance of the problem, you program should print $n-1$ lines, describing a sequence of contractions that transforms the starting sequence into the number T corresponding to that instance. Each of these $n-1$ lines contains just an integer i, denoting the ith contraction to be applied.

You can assume that at least one such sequence of contractions will exist for a given input. Each instance should be terminated with a blank line.

Sample Input

```
4 5
10
2
5
2
5 4
12
10
4
3
5
0 0
```

Sample Output

```
1
2
1

2
3
2
1
```

PROBLEM 7: Keyless Encryption(?!)

File Names

Source File: crypto.pas or crypto.c or crypto.cpp
Input File: crypto.in
Output File: crypto.out

Statement of the Problem

The chairman of "Security Inc." has decided that all messages exchanged by the staff must be encrypted. After some research, one of the employees came up with a simple algorithm: it requires first the construction of a list $A_1, A_2, \ldots, A_n$ of all the words in the message, including punctuation marks, in order of their occurrence in the message. After that, the algorithm reads the list sequentially doing the following: if the word A_j has a previous duplicate A_i such that there is not another duplicate of A_j between them, then the algorithm swaps the sublist A_1, …, A_{i-1} with the sublist A_{i+1}, …, A_{j-1}, and so on. For example, if we have the message "a binary tree is a tree", then the algorithm generates the following steps:

`a`		`binary`		`is`
`binary`		`tree`		`a`
`tree`	$\Longrightarrow$	`is`	$\Longrightarrow$	`a`
`is`		`a`		`tree`
`a`		`a`		`binary`
`tree`		`tree`		`tree`

Write a program that takes a encrypted message as input and generates the original decrypted message.

Input Format

The input file may contain several instances of the problem. A line containing the character % terminates each instance, except the last one, which is terminated by the character #. These symbols never appear as part of the messages themselves

Each instance consists of a message in one or more lines, that is, a sequence of words (lower and upper-case letters) and punctuation marks.

Output Format

For each instance of the problem, your program should print the original decrypted message. Each instance is terminated with a blank line.

Sample Input

```
is a a tree binary tree
%
or not to to be be
#
```

Sample Output

```
a binary tree is a tree

to be or not to be
```