ACM South American Regional

Collegiate Programming Contest

Contest Session

November 18, 2000

Buenos Aires – Argentina
Campina Grande – Brazil
Porto Alegre – Brazil
São Paulo – Brazil
Santiago – Chile
Merida – Venezuela

acm International Collegiate Programming Contest 2001 IBM event sponsor
CREDITS HOME REGIONALS WORLD FINALS INFO PAST CONTACTS

PROBLEM 1: S-Trees

File Names

Source File: stree.c, etc
Input File: stree.in
Output File: stree.out

Statement of the Problem

A Strange Tree (S-tree) over the variable set $X_n = \{x_1, ..., x_n\}$ is a **complete** binary tree representing a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. Each of the S-tree's nodes has a depth, defined as the number of nodes in the path from the root to itself minus 1 (so the root has depth 0). The depth of any node in an S-tree is at most n. The nodes with depth less than n are called *non-terminal nodes*, each having two children: the right child and the left child. Each non-terminal node is labeled with some variable x_i from the variable set X_n. All non-terminal nodes with the same depth are labeled with the same variable, and non-terminal nodes with different depth are labeled with different variables. So, there is a unique variable x_{i_0} corresponding to the root, a unique variable x_{i_1} corresponding to the nodes with depth 1, and so on. The sequence of the variables $x_{i_0}, x_{i_1}, ..., x_{i_{n-1}}$ is called the variable ordering. The nodes having depth n are called *terminal nodes*. They have no children and are labeled with either 0 or 1. Note that the variable ordering and the distribution of 0's and 1's on terminal nodes are sufficient to completely describe an S-tree.

As stated earlier, each S-tree represents a Boolean function f. If you have an S-tree and values for the variables $x_1, ..., x_n$, then it is quite simple to evaluate $f(x_1, ..., x_n)$: Start at the root and repeat the following steps: if the node you are at is labeled with a variable x_i, then depending on whether the value of x_i is 1 or 0, you go to its right or left child, respectively. Once you reach a terminal node, its label gives the value of the function.

```
              x1                                   x3
          /       \                            /       \
       /             \                      /            \
      x2              x2                   x1              x1
    /    \          /     \             /     \         /      \
  x3      x3      x3      x3           x2      x2      x2      x2
 /  \    /  \    /  \    /  \         /  \    /  \    /  \    /  \
0     0 0     0 0     1 1     1      0     0 0     1 0     0 1     1
```

Fig. 1.- S-trees for the function x_1 *and* $(x_2$ *or* $x_3)$

In the picture, the two S-trees represent the same Boolean function:

$$f(x_1, x_2, x_3) = x_1 \textit{ and } (x_2 \textit{ or } x_3) .$$

For the left tree, the variable ordering is x_1, x_2, x_3, and for the right tree it is x_3, x_1, x_2. The values of the variables $x_1, ..., x_n$, are given as a Variable Value Assignment (VVA) $(x_1 = b_1, x_2 = b_2, ..., x_n = b_n)$ with $b_1, b_2, ..., b_n \in \{0,1\}$. For instance, $(x_1 = 1, x_2 = 1, x_3 = 0)$ would be a VVA for $n = 3$, resulting, for the sample function above, in the value $f(1,1,0) = 1$ *and* $(1$ *or* $0) = 1$. Your task is to write a program which takes an S-tree and some VVAs and computes $f(x_1, ..., x_n)$ as described above.

Input Format

The input file contains the description of several S-trees with associated VVAs which you have to process. Each description begins with a line containing a single integer $n, 1 \leq n \leq 7$, the depth of the S-tree. This is followed by a line describing the variable ordering of the S-tree. The format of that line is $x_{i_1} x_{i_2} ... x_{i_n}$. (There will be exactly n different single space-separated strings). So, for $n = 3$ and the variable ordering x_3, x_1, x_2, this line would look as follows

```
x3 x1 x2
```

In the next line the distribution of 0's and 1's over the terminal nodes is given. There will be exactly 2^n characters (each of which can be 0 or 1), followed by the new-line character. The characters are given in the order in which they appear in the S-tree, the first character corresponds to the leftmost terminal node of the S-tree, the last one to its rightmost terminal node.

The next line contains a single integer m, the number of VVAs, followed by m lines describing them. Each of the m lines contains exactly n characters (each of which can be 0 or 1), followed by a new-line character. Regardless of the variable ordering of the S-tree, the first character always describes the value of x_1, the second character describes the value of x_2, and so on. So, the line

```
110
```

corresponds to the VVA $(x_1 = 1, x_2 = 1, x_3 = 0)$.

The input is terminated by a test case starting with $n = 0$. This test case should not be processed.

Output Format

For each S-tree, output the line "S-Tree #j", where j is the number of the S-tree. Then print a line that contains the value of $f(x_1, x_2, ..., x_n)$ for each of the given m VVAs, where f is the function defined by the S-tree. Output a blank line after each test case.

Sample Input

```
3
x1 x2 x3
00000111
4
000
010
111
110
3
x3 x1 x2
00010011
```

```
4
000
010
111
110
0
```

Sample Output

```
S-Tree #1
0011

S-Tree #2
0011
```

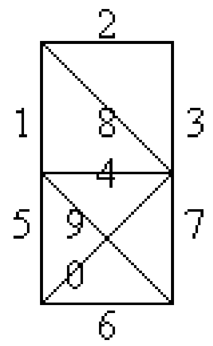
PROBLEM 2: Led Codes

File Names

Source File: ledcode.c etc
Input File: ledcode.in
Output File: ledcode.out

Statement of the Problem

Some led systems display characters as a combination of lightlines, much like digital calculators do. Suppose we have one of such led systems, in which each letter of the standard English alphabet is constructed combining some of ten possible lightlines, numbered as below:



For instance, the letter A is shown using the lines 1,2,3,4,5 and 7. With these few lines it is not possible, of course, to show all the 26 letters with their natural shapes. A complete list of the letters of our system is appended at the end.

You have to solve the problem of decoding a combination of numbers (representing lightlines) in the form of the respective letter, to form a whole word or phrase. As input you have a string of letters and/or numbers, and as output you must show only the letters.

Input Format

The input file begins with a line containing a single integer N, that indicates the number of input examples, followed by N lines, each one representing an input phrase. Each input phrase consists of lowercase letters, blank spaces and/or digits. In the case of digits, their combination must form valid led letters. Each letter is coded as a combination of numbers, ordered in the form 1,2,3,4,5,6,7,8,9,0. A zero (0) that is not part of a valid letter code, is interpreted as a blank space. You may assume that no invalid codes are entered, and that the system does not allow ambiguity between two letter codes.

Output Format

The output file consists of N lines (one for each input string) with the form:

```
Phrase 1: .....
Phrase 2: .....
```

where the dots are, of course, the result of the decoding process. The alphabetic letters and blank spaces must not be decoded at all: only the numbers must be converted to the corresponding led letters.

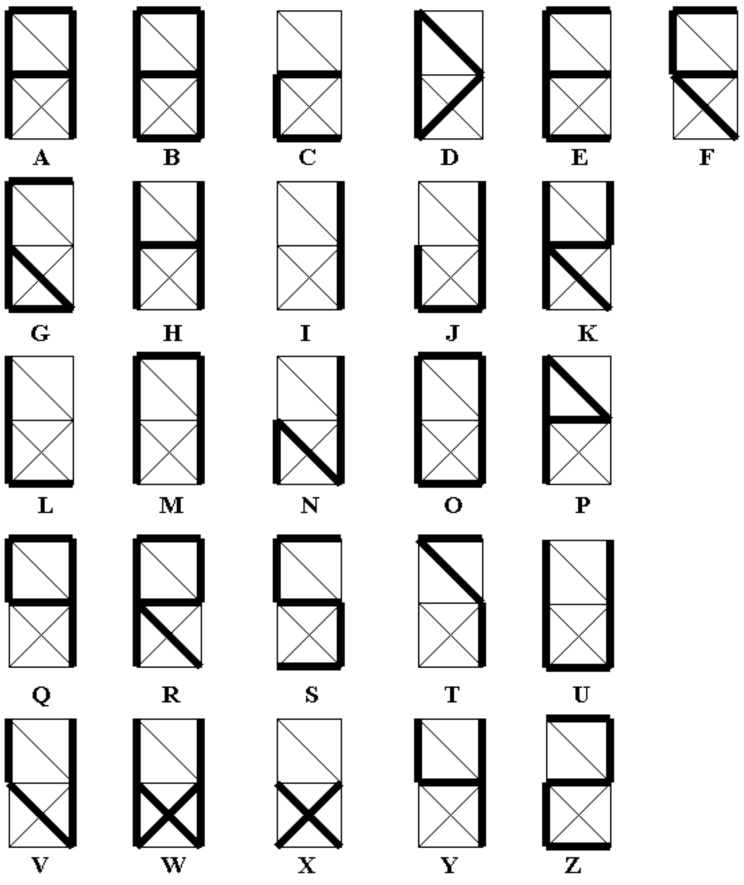
Sample Input

```
3
HELL1235670W01234591561580
PROGRAMMING037124670C123567123567156
AND MORE0371249037357902781345737124670456123567357927812456124672278
```

Sample Output

```
Phrase 1: HELLO WORLD
Phrase 2: PROGRAMMING IS COOL
Phrase 3: AND MORE IF IN THIS CONTEST
```

Complete List of Letters



PROBLEM 3: Cable Network

File Names

Source File: cn.c etc
Input File: cn.in
Output File: cn.out

Statement of the Problem

A local cable network is planning to expand their services to a neighboring city. For that, a transmission center is being built and a cable net connecting the new clients has to be installed. Each client has to be connected to the center in the cable net.

The network organized a survey and has now the addresses of all potential clients. Also it has an estimate on the cost of installing each cable link through the city.

The network board of directors has decided to offer the following initial subscription package: Each new client will pay a fixed subscription fee and will get the first three months of service for free. This ensures a **potential initial capital** for building the cable net, which is simply the number of potential clients (according to the survey) multiplied by the subscription fee.

The decision on what links to build in this first phase will be done according to the following. A set of links induces a **loss in the potential initial capital**, defined as $l + (t \times d)$, where l is the cost of installing the links in the set, t is the subscription fee and d is the number of potential clients that are **not** connected to the center by this set of links. The board of directors decided they will build a set of links that induces a loss as small as possible in the potential initial capital.

Your job is to write a program to compute the minimum possible loss in the potential initial capital. For that, you are given the location of each potential client as well as the center location. Also, you are given the costs of all possible net links and the value of the subscription fee.

Input Format

The input file contains the description of several instances. An instance with $n = 0$ indicates the end of the input. Your program should not process this instance.

The first line of the file, contains two positive integers: n $(0 \leq n \leq 20)$ representing the number of potential clients and t representing the subscription fee. Each potential client is identified by a number between 1 and n while the center is identified by 0. Next, there is a line for each i from 0 to $n-1$. In line i there is a list of $n-i$ positive integers. Each integer, say the j-th one, represents the cost of installing a link between i and j.

Output Format

For each instance, your program should print three lines; the first contains "Cable Net #j", where j is the number of the instance; in the second line, it should print the potential initial capital for that instance and in the third line, the value of the minimum possible loss in the potential initial capital for that instance. Your program should also print a blank line after the output of each instance.

Sample Input

```
5 10
5 17 2 3 23
9 6 1 19
14 5 13
19 21
40
3 1
5 3 23
1 19
14
0 1
```

Sample Output

```
Cable Net #1
50
21

Cable Net #2
3
3
```

PROBLEM 4: Bomb the bridge

File Names

Source File: bridge.c etc
Input File: bridge.in
Output File: bridge.out

Statement of the Problem

The Air Force has a special team devoted to destroying bridges. This team is composed of several planes which fly over the targeted bridge one after another. Each of these planes carries one bomb, which the pilot drops at a given location in the bridge. The aim of the team is to split the bridge in two, so that no one can cross it.

Unfortunately, bombs are not completely accurate. As a result, even though missions are planned in advance, the exact location where the bombs actually fall may be different than planned. Still, the Air Force keeps his original plan untouched except for the last plane, which may have to act differently.

Each bomb that is dropped on the bridge leaves a hole that, viewed from above, has the shape of a circle, its radius depending on the power of the bomb. A bomb of size s leaves a hole of radius s. Your task is to determine the plan for the last plane, that is, the minimum bomb size and the position where the last plane should drop the bomb so that, if everything goes well, the bridge is split in two.

Input Format

The first line contains a positive integer n, representing the number of instances of the problem contained on the input file.

Each instance is represented by three positive integers w, ℓ and b, which denote the width and length of the bridge and the amount of bombs dropped on the bridge. Then come b lines, each containing integers x, y and r $(x, y \geq 0, r > 0)$ which denote the coordinates of the center and the radius of the holes left by the b bombs. Coordinate (0,0) is the lower-left corner of the bridge (viewed from above), assuming that the bridge has width parallel to the x axis and length parallel to the y axis.

Output Format

Your program must produce one output line for each instance. This line contains an **integer** representing the minimum bomb size you need to split the bridge in two, or the words `Bridge already split` if the bridge is already split by the bombs that have been dropped. The bridge is considered to be split if the two ends of the bridge are disconnected (except perhaps for a finite number of points). Remember that a bomb of size s leaves a hole of radius s.

Sample Input

```
3
100 1000 2
15 100 20
90 100 30
100 1000 1
50 100 40
100 1000 2
25 500 25
75 500 25
```

Sample Output

```
13
50
Bridge already split
```

PROBLEM 5: Move the Water

File Names

Source File: water.c etc
Input File: water.in
Output File: water.out

Statement of the Problem

You have three jars containing water. Each jar has associated a capacity indicating the maximum amount of water it can contain. Your wish is to have a certain amount of water in each jar. The initial content of each jar, its capacity, and the desired content, are all non negative integers; in addition, the capacities are between 1 and 150. You have nothing but the jars to help you in achieving your goal. So the only way to obtain the desired content in each jar is to make a sequence of *movements* of water. In a movement of water you pour a certain amount of water from one jar to another jar. We call the first jar *source* and the second jar *destination*. While moving water two situations can occur: If the current content of the source jar is not enough to complete the capacity of the destination jar, then all the water contained in the source jar is moved to the destination jar; otherwise, only the exact amount of water needed to complete the destination jar is moved from the source jar. Under no circumstances you can use additional water or throw away water. Your task is to determine the minimum number of movements required to obtain the desired amount of water in each jar.

Input Format

The input file contains a certain number of test cases for this problem. Each test case is given in a single line containing nine values c_1, c_2, c_3, a_1, a_2, a_3, b_1, b_2, b_3; these values are separated by an arbitrary number of blank spaces. The value c_i is the capacity of the i-th jar, a_i is its initial content, and b_i is the desired content for the jar. In the last line of the file c_1 has value equal to 0; this line should not be processed.

Output Format

For each test case in the input file, the output file must contain a line with the minimum number of movements required to obtain the desired content in each jar. If for a given test case there is no solution, you must write the value -1 instead of the number of movements.

Sample Input and Output

Input

```
5 4 1   1 0 1   0 2 0
5 4 1   1 0 1   0 3 0
5 4 1   1 0 1   0 0 2
5 4 1   1 0 1   1 0 1
5 4 1   0 4 0   3 0 1
0 1 1   0 0 0   0 0 0
```

Output

```
2
-1
-1
0
2
```

PROBLEM 6: Faking Mondrian

File Names

Source File: mondrian.c etc
Input File: mondrian.in
Output File: mondrian.out

Statement of the Problem

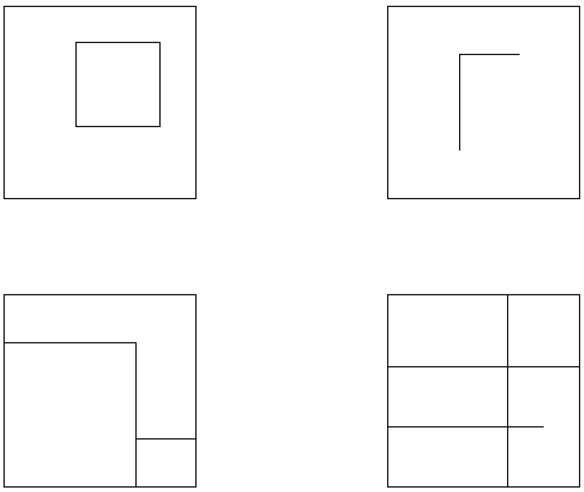
Piet Mondrian (1872-1944) was one of the best abstract artists of the century. His simple yet elegant designs were intentionally restricted to use only straight lines and a few colors. His artwork reflects an idealized view of reality, one which has perfect harmony and balance.

Armando, a Latin American businessman, wants to create and sell fake Mondrian pictures. Since Mondrian's designs are based on a few simple rules, he hopes to build a machine that can mass-produce different Mondrian pictures and sell them as originals.

Mondrian's rules are very simple:

1. only black, white, red, blue and yellow colors can be used.

2. the painting can only contain rectangles, with sides parallel to the borders of the canvas. Lines that are not part of the border of a rectangle are not allowed. The following four examples do not fulfill this rule:



The top-left painting contains a shape that is not a rectangle (the shape that is outside the central square). In the top-right painting every shape is a rectangle, but there are lines that are not part of the border of any rectangle. Bottom-left painting contains a shape that is not a rectangle (the inverted L). Bottom-right painting contains a line that is not part of the border of any rectangle. In the figure we also include a real (well, sort of) Mondrian (Composition with red, yellow, and blue, 1928).

3. Borders of the rectangles can only be painted in black. Rectangles can only be painted in white, red, blue or yellow. The borders of the canvas are considered to be already painted in black.

4. Two adjacent rectangles (that is, two rectangles that share a segment of their borders) cannot have the same color, unless they are painted in white.

The machine Armando has devised receives a set of instructions to draw the painting. Your task is to program an algorithm that will tell (before the machine begins to perform Armando's instructions) whether these instructions correspond to a picture that fulfills Mondrian's rules, and how much paint he will need.

Input Format

The input file contains a set of test cases. Each test case is described in several lines, and contains Armando's instructions for one painting. There are two types of instructions: instructions that indicate the drawing of a line, and instructions that indicate the painting of a rectangle. Each test case of the input file starts with a line containing four integer values w, h, n, m. The values w and h represent the width and height of the canvas, respectively; these values are between 1 and 10000. The value n is the number of instructions for drawing lines, while m is the number of instructions for painting rectangles; these values are between 0 and 1000. The test case continues with n lines containing the instructions for drawing lines, each instruction in a separate line of the file. Each instruction consists of four integer values x_1, x_2, x_3 and x_4, where (x_1, x_2) and (x_3, x_4) are the coordinates of the end points of the line. Lines are always horizontal or vertical. The lower left corner of the canvas has coordinates (0,0). Instructions can result in a line or part of a line being drawn more than once. The test case ends with m lines containing the instructions for painting rectangles, each one in a separate line of the input file. Each instruction specifies two integer values x and y, as well as a letter c in the set $\{r, b, y\}$ (r means red, b is blue and y is yellow). The instruction indicates the machine to paint the rectangle containing the point with coordinates (x, y) with color c. All rectangles not painted with a color in this set are assumed to be painted in white. The input file ends with a test case in which w has a value of 0; this test case should not be processed. Every pair of consecutive values in a same line of the input file is separated by an arbitrary number of blank spaces.

Output Format

Your program must produce one output line for each test case. If the instructions follow Mondrian's rules, you must write the line `Mondrian r b y`, where r, b and y are total areas painted in red, blue and yellow, respectively. Otherwise, you must write the line `Error!`.

Sample Input

```
100 100 3 3
0 50 60 50
20 50 100 50
50 0 50 100
25 25 r
```

```
75 75 r
25 75 y
610 987 2 3
0 50 305 50
305 0 305 987
1 1 r
1 400 r
400 100 y
1000 1000 3 2
0 200 500 200
500 0 500 400
500 400 1000 400
200 800 r
700 200 y
0 0 0 0
```

Sample Output

```
Mondrian 5000 0 2500
Error!
Error!
```