1999 APICS Programming Contest Problems

The 1999 Atlantic Canada programming Contest was held at Memorial University in St John's NF Friday Oct 22 1999. There was 1 team of 3 students from each of nine universities in the four Atlantic provinces.

Each team of 3 students was given 6 problems to try to solve in 5 hours using one computer following the rules for the ACM programming contests.

Each problem has sets of associated data files and matching output files. The files beginning with a lowercase letter are available to the contestants during the competition. The others are reserved for the judges to test submitted solutions.

The participating teams came from the folowing universities:

```
University of New Brunswick, Fredericton Campus
University de Moncton
Mount Allison Univ., Sackville N.B.

Acadia Univ., Wolfville N.S.
Dalhousie Univ., Halifax N.S.
St Marys Univ., Halifax  N.S.
St Francis Xavier Univ., Antigonish N.S.

Univ. of Prince Edward Island.

Memorial Univ., Newfoundland and Labrador.
```

The top two teams from this competition went to the next level of competition, Nov. 6 in Westfield Massachusetts.

Switching (Patricia Evans)

Balls are being dropped into a machine that contains 4 switches. Four bins are at the bottom of the machine for the balls to fall into. The top switch is switch 1, which the balls will hit first. If the switch is to the left, it will lead to switch 2; if to the right, it will lead to switch 3. Switch 2 to the left leads to bin 1, and to the right leads to switch 4. Switch 3 to the right leads to bin 4, and to the left leads to switch 4. Switch 4 to the left leads to bin 2, while to the right it leads to bin 3.

When a ball hits a switch, it passes through it in its current direction, but the switch then flips (from left to right or right to left) to point the other way. Balls pass through the machine one at a time.

Input: you are given the initial configuration of the 4 switches, in a ordered list <s1,s2,s3,s4> where s1 represents switch 1, s2 represents switch 2, etc. Each s# is a word, either left or right (case-insensitive). The following line contains the number n of balls that pass through the machine.

Sample Input:(a1.dat)
<left, LEft, right, RIGHT>
3

Output: The output should give the number of balls in each bin (with the bins labeled). Sample Output: (a1.out)
Bin 1: 1
Bin 2: 0
Bin 3: 1
Bin 4: 1

Text Justification (A. Sedgwick)

In the dark ages printers only had fixed-width fonts where every character had the same width when printed. A common problem was to take input text and format it so that each line had the same number of characters. This was done by joining lines of text together and putting extra blanks between words where necessary.

Write a program which takes input text and prints it in justified form as shown in the example below. Each output line has a certain WIDTH and a certain INDENT. Unless otherwise specified the WIDTH should be 40 and the INDENT 0. All output lines should begin with a number of blanks equal to the current value of INDENT.

The input consists of any number of lines each consisting of any number of tokens. The tokens are separated by the usual "whitespace" characters. Some tokens are treated specially. "<P>" signals a new paragraph. "<W65>" changes the WIDTH to 65. The number may be any positive integer not bigger than the maximun value allowed by the operating system. Tokens like "<I12>" are used to change the INDENT for subsequent output to the value given, in this case, 12.

All lines except the last in a paragraph should be "filled" and "justified". This means that they should have as many tokens as possible and that the number of blanks between tokens is adjusted so that the number of characters in the line is exactly WIDTH and the number of blanks between tokens is as even as possible.

The last line in a paragraph may be shorter. It should have only one blank between words.

In the output all paragraphs are separated by empty lines. If there are several <P> tokens in a row there should be the same number of blank lines before the next paragraph of output.

If a token is longer than WIDTH - INDENT it should be output on a line by itself.

```
<Sample Input b1.dat>
<W15>
This  is a  line too
long,
and this   is
short:
<P> <I3> To be
or not
to be. <P>
 That is the question!

<Corresponding Output b1.out>
This is a  line
too  long,  and
this is short:

   To be or not
   to be.

   That is the
   question!
```

There is a second data file b2.dat with matching output that you may find helpfull.

The Stable Marriage Problem. (M. McAllister)

There are n men and n women in a monogamous (heterosexual) society. Each person has a list of preferences for the members of the other sex. The problem is to find a way of pairing each man with a woman in a stable way. A pairing is stable if you can not find any man and a woman who would prefer each other to the people they have been paired with.

The input to the program gives the number of couples followed by the preferences of the woman and then the preferences of the men. For example:

```
4
3 2 1 4
4 1 2 3
3 1 2 4
1 4 3 2
2 3 1 4
4 3 1 2
2 3 1 4
4 2 1 3
```

The first line contains the value of n. The next n lines contain the preferences of the women. Woman 1 prefers the men in the order 3, and then 2 then 1 and finally 4. The last n lines contain the men's preferences. In this example both man 1 and man 3 rank the women in the order 2 first, 3 second and then 1 followed by 4.

An example of an unstable pairing would be to match man 1 with woman 3 and man 2 with woman 2. Then man 1 and women 2 would be tempted to elope since they both would prefer each other to the people they have been assigned.

The output for the above input might be:

```
1 2
2 1
3 3
4 4
```

This means that woman 1 is paired with man 2 and woman 2 with man 1 etc.

Pairing is a very important process. We might pair students with employers or students applying to universities. You might expect to use a "greedy" algorithm to solve this problem but this can lead to failure. Gale and Shapley found an algorithm that always works in 1962. The key is to allow only one sex to propose!

The algorithm is a sequence of proposals: men propose and women either accept or reject. At a stage of the algorithm, some man currently unpaired proposes to the most desirable woman among those who have not rejected him. If the woman is currently unpaired then she accepts. If the woman is currently paired then if the new proposal is better than her current one then she accepts him and rejects the current partner - if the new proposal is worse than her current one then she rejects the new proposal. If you prefer you can have the woman propose to the men but only one gender can propose.

Write a program to read the preference data and output a stable pairing. The above data is available in file c1.dat. You can run the with input redirected from this file. There are other test data sets c2.dat and c3.dat. The judge has other data sets and a program to test if your output represents a stable pairing.

The permutation problem (P. Scobey)

An input file contains two integers on its first and only line, the second integer greater than or equal to the first. Write a program that will find, and display on the standard output, one per line, all integers that are formed by permuting the digits of the second integer and that are also divisible by the first integer. In addition, the values output must be displayed in ascending order.

Sample input (available for redirection as file d1.dat)
1 111

Corresponding output (available as d1.out)
111

Another sample input (d2.dat)
123 123321

Corresponding output (available as d2.out)
123123
323121

Another sample input (d3.dat)
11 4321

Corresponding output (available as d3.out)
1243
1342
2134
2431
3124
3421
4213
4312

Detective Dodgson (J. Horton)

Detective Dodgson is trying to solve a mystery. He is,
however, rather disorganized, and has a folder of clues in
an apparently random order. Each clue is either a basic sentence
or, if it contains the word "implies", is put together
from two other basic sentences. The last line of the file
of clues contains a question, of the form "How do we know that ... ?"
(where the ... is replaced by a basic sentence).

Dodgson understands his clues, but he has called in sick
and it is left to you to use his clues to solve the mystery
and make a case for your answer. Your case is to consist
of a list of the sentences that give you your deduction
(first a basic, then an implies, then a basic, then implies, etc.)
finishing with the basic sentence that is to be deduced.
This list should be as short as possible.
If there is no way to derive the conclusion from the facts,
answer with "We don't.".

Note that the initial letter of each clue will be capitalized,
and it will end with a period.

Sample Input: (available as file e1.dat)
The wheat is white.
The wheat grows in circles implies the aliens have landed.
The wheat is white implies the wheat has not been cut.
The wheat is white implies the wheat grows in circles.
How do we know that the aliens have landed?

Sample Output: (available as file e1.out)
The wheat is white.
The wheat is white implies the wheat grows in circles.
The wheat grows in circles.
The wheat grows in circles implies the aliens have landed.
The aliens have landed.

For another example see files e2.dat and e2.out.

Layers of Communication (P. Evans)

Some number n of machines are arranged in order (1 through n) around a circle. Communication links are to be installed between specified pairs of machines. All links are straight, and must be within the circle of machines. However, we want to organize these links into two layers, where two links cannot be part of the same layer if they cross each other. Being attached to the same machine does not count as crossing.

Given the number n and the pairs of machines that are to be linked, organize the links into two layers (if possible). If the organization is possible, print out the list of links in each layer. If it is not possible, print out "not possible".

The input file will have on its first line the integer n. Each remaining line is a pair of numbers (from 1 through n) that give the pairs of machines to be linked, in no particular order. Note that there may be many possible different organizations, and many ways to list them.

Sample Input (available as file f1.dat):

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

Sample Output:

```
Layer 1:
2 4
2 5
1 5
4 5

Layer 2:
1 3
1 4
3 4
```

Another data file for which a solution is not possible is available as file f2.dat.