Waterloo ACM Programming Contest

September 25, 2004

Problem A: Practice

How much does winning ACM depend on practice?

We assume that *p*, the probability that a given team will win a given contest, is related to *n*, the number of practice problems solved by the team prior to the contest. This relationship is modelled by the *logistic* formula

$$log(p/(1-p)) = a + b\ n\ ,$$

for some *a* and *b*. Your job is to find *a* and *b* such that the formula most accurately reflects a set of observed results.

Each observation consists of *n* and *w*. *n* is the number of practice problems solved by some team prior to a contest, and *w* is 1 if the team wins the contest, 0 if it does not.

Given *a*, *b*, and *n* the formula above may be used to compute *p*, the estimated probability that $w = 1$. The *likelihood* of a particular observation is *p* if $w = 1$ and *1-p* if $w = 0$; The likelihood of a set of observations is the product of the likelihoods of the individual observations.

You are to compute the *maximum likelihood estimate* for *a* and *b*. That is, the values of *a* and *b* for which the likelihood of a given set of observations is maximized.

The input contains several test cases followed by a line contatining 0. Each test case begins with $1 < k \leq 100$, the number of observations that follow. Each observation consists of integers $0 \leq n \leq 100$ and $0 \leq w \leq 1$. The input will contain at least two distinct values of *n* and of *w*. For each test case, output a single line containing *a* and *b*, rounded to four digits to the right of the decimal.

Sample Input

```
20
0 0
0 0
0 0
0 0
1 0
1 0
1 0
1 1
2 0
2 0
2 1
2 1
3 0
3 1
3 1
3 1
4 1
4 1
4 1
4 1
0
```

Output for Sample Input

```
-3.1748 1.5874
```

Gordon V. Cormack

Problem B: Help!

MegaFirm Inc. has created a set of patterns to aid its telephone help-desk operators in responding to customers. A pattern is a phrase consisting of words and placeholders. A word is simply a string of letters. A placeholder is a word enclosed in angle brackets (that is < ... >). A phrase *matches* a pattern if each placeholder in the pattern can be systematically replaced by a word so as to make the pattern and phrase equal. By "systematically replaced" we mean that all placeholders enclosing the same word are replaced by the same word.

For example, the phrase

```
to be or not to be
```

matches the pattern

```
<foo> be <bar> not <foo> <baf>
```

because we can replace <foo> by to, <bar> by or, and <baf> by be.

Given two patterns, you are to find a phrase that matches both.

The first line of input contains *n*, the number of test cases. Each test case consists of two lines of input; each a pattern. Patterns consist of lowercase words, and placeholders containing lowercase words. No pattern exceeds 100 characters. Words contain at most 16 characters. A single space separates adjacent words and placeholders.

For each test case, output a phrase that matches both patterns. If several phrases match, any will do. If no phrase matches, output a line containing "-" (a single minus sign).

Sample Input

```
3
how now brown <animal>
<foo> now <color> cow
who are you
<a> <b> <a>
<a> b
c <a>
```

Possible Output for Sample Input

```
how now brown cow
-
c b
```

Gordon V. Cormack

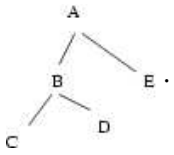
Problem C: Treequivalence

The following grammar describes a textual notation for a tree with (not necessarily unique) vertex labels:

```
tree ::= label
tree ::= label ( subtrees )
subtrees ::= tree
subtrees ::= subtrees , tree
label ::= A | B | C | ... | Z
```

That is, the representation of a tree consists of a label (which is an uppercase letter) or a label followed by a bracketed ordered list of trees separated by commas.

In order to draw such a tree on paper, we must write each label on the page, such that the labels for the subtrees of a vertex are positioned counter-clockwise about the vertex. The labels must be positioned such that non-intersecting line segments connect each vertex to each of its subtrees. That is to say, we draw the normal planar representation of the tree, preserving the order of subtrees. Beyond these constraints, the position, shape, and size of the representation is arbitrary.

For example, a possible graphical representation for A(B(C,D),E) is 

.

Given the textual representation for two trees, you are to determine whether or not they are equivalent. That is, do they share a common paper representation?

The first line of input contains *t*, the number of test cases. Each test case consists of two lines, each specifying a tree in the notation described above. Each line will contain at most 200 characters, and no white space. For each test case, output a line containing "same" or "different" as appropriate.

Sample Input

```
2
A(B(C,D),E)
E(A,B(C,D))
A(B(C,D),E)
E(A(B(C,D)))
```

Output for Sample Input

```
different
same
```

Gordon V. Cormack

Problem D: Antiarithmetic?

A permutation of n is a bijective function of the initial n natural numbers: 0, 1, ... n-1. A permutation p is called antiarithmetic if there is no subsequence of it forming an arithmetic progression of length bigger than 2, i.e. there are no three indices $0 \leq i < j < k < n$ such that (p_i , p_j , p_k) forms an arithmetic progression.

For example, the sequence (2, 0, 1, 4, 3) is an antiarithmetic permutation of 5. The sequence (0, 5, 4, 3, 1, 2) is not an antiarithmetic permutation as its first, fifth and sixth term (0, 1, 2) form an arithmetic progression; and so do its second, forth and fifth term (5, 3, 1).

Your task is to check whether a given permutation of n is antiarithmetic.

There are several test cases, followed by a line containing 0. Each test case is a line of the input file containing a natural number $3 \leq n \leq 10000$ followed by a colon and then followed by n distinct numbers separated by whitespace. All n numbers are natural numbers smaller than n.

For each test case output one line with `yes` or `no` stating whether the permutation is antiarithmetic or not.

Sample input

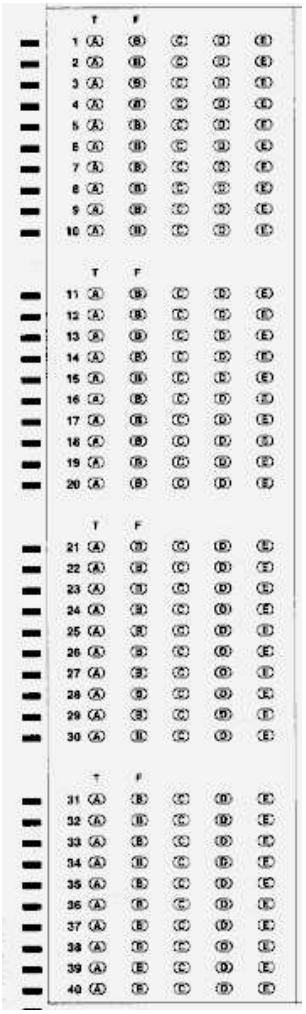
```
3: 0 2 1
5: 2 0 1 3 4
6: 2 4 3 5 0 1
0
```

Output for sample input

```
yes
no
yes
```

Piotr Rudnicki

Problem E: Test



A vocational preference test, unlike an aptitude test, seeks to identify careers that the candidate might find satisfying. Based on the answers to a slew of seemingly inane multiple-choice questions like the one below, the test determines which occupations suit the candidate's personality.

```
Which would you rather spend an afternoon doing?

(a) feeding chickens
(b) driving a race car
(c) watching The Simpsons on TV
(d) suntanning
(e) building a dog house
```

Each question asks the candidate to express a preference from among five activities, selected from a common larger set. That is, activities like *feeding chickens* or *suntanning* are likely to appear in several different questions.

If a candidate answers A in a question containing A, B, C, D, E as alternatives, this choice indicates a preference for A over each of B, C, D, E. Also, if one answer indicates a preference for X over Y and one or more other answers indicate a preference for Y over Z, the combined set of answers indicates a preference for X over Z.

The candidate may provide contradictory answers; that is, the answers may indicate a preference for X over Y and also for Y over X. These contradictions indicate inconsistency, a personalitiy attribute that may suggest a career in politics or used auto sales.

Given a set of answers to a vocational preference test, you are to partition the activities into the minimal number of sets such that, for each pair within a given set, the answers indicate a contradictory preference.

The input contains several test cases followed by a line containing 0. Each case begins with *n*, the number of questions in the test. *n* lines follow, each containing the names of five distinct activities, followed by the candidate's answer - one of the five alternatives. Each activity is named by a single upper case letter.

For each test case, output the sets, one per line. Output the elements of each set in alphabetical order, and output the sets in alphabetical order by their least element. The sets should together contain exactly the set of activities that appear in the input. Leave an empty line between test cases.

Sample Input

```
4
A B C D E C
F C H I J J
K B H I F I
K C E B J K
0
```

Output for Sample Input

```
A
B
C
D
E
F
H
I J K
```

Gordon V. Cormack