Twelfth Annual
University of Oregon Programming Competition

Saturday, February 23, 2008

Supported in part by Microsoft

Problem Contributors

Jim Allen

David Atkins

Art Farley

Aaron Henner

Eugene Luks

Matt Sottile

DEAL $\vee$ ¬DEAL

Deal or No Deal is a popular TV game show in many countries. It is essentially a game of the psychology of gambling where a player has a chance to win an amount of money or be paid a guaranteed sum to give up that chance.

In the US version, the game begins with 26 cases that hold various monetary figures, from one cent to one million dollars. The player chooses one case which is theirs to keep, but is not told its contents. Then the player picks six of the remaining cases, and their amounts are revealed. The banker makes an offer to buy the player's case. If the player takes the money (Deal), then the game is over and the player gets the amount offered, regardless of what was in the chosen case. If the player turns down the offer (No Deal), then in the next round five more cases must be picked and revealed, and a new offer will be made. With each successive round, one less case is picked until the cases are chosen one at a time.

$ 0.01	$ 1,000
$ 1	$ 5,000
$ 5	$ 10,000
$ 10	$ 25,000
$ 25	$ 50,000
$ 50	$ 75,000
$ 75	$ 100,000
$ 100	$ 200,000
$ 200	$ 300,000
$ 300	$ 400,000
$ 400	$ 500,000
$ 500	$ 750,000
$ 750	$ 1,000,000

Psychology enters when player must decide whether to turn down the deal and continue to choose cases. The only way for the player to get the amount in her case is to turn down all offers. Of course, if the higher amounts are revealed, then the player's case cannot hold those values, and subsequent offers will decrease. From a probabilistic point of view, the expected value at any point in the game is simply the average of the amounts that have not been revealed. If everything was even, the banker would offer the expected value at each point, but in the game show, the banker usually offers somewhat less than this amount, trying to buy it for less than its expected value.

In this problem, you are given a sequence of game configurations. Each configuration will consist of a number of cases to be revealed, and a list of the values in all the cases remaining. For each configuration, you are to print out "`Deal`" if the probability that the next offer will be less than the current offer is less than 1/2. Print out "`No Deal`" if it is at least twice as likely that the next offer will exceed the current. If neither of these conditions holds, then print out "`Feeling lucky?`". For any game configuration, assume that the banker will actually offer the expected value, i.e., the average of the remaining cases (including the original one chosen).

The first line of input will be a number that specifies how many configurations are to be evaluated. Each configuration will be on a single line and consist of an integer from 1 to 6, followed by amounts as in the table. All values are separated by spaces. Your output must consist of the messages above, with the message for each configuration on a line.

Sample Input

```
4
1 1000000 500000 10
1 75000 50000 5
1 .01 1000000
3 25000 1000 750 500 400 300 200 50 5 1 .01
```

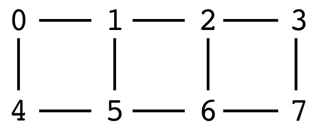
Sample Output

```
No Deal
Deal
Feeling lucky?
No Deal
```

MESSAGE MONITORS

Suppose we want to monitor messages being sent between two sites of a network without the two sites knowing we are doing so. We could do this by setting up a special “listening” process at some *other* site in the network. We assume that messages (or pieces of messages) always follow a shortest path between two sites in the network. If there is more than one shortest path, then any of these paths could be used at any time. Thus, to monitor all messages effectively, we must find a site that is on *all shortest paths between the two sites of interest*, if such sites exist. Assume that every network is connected, i.e., there is at least one (shortest) path between every pair of vertices.

The graph of a network is input as one line indicating how many vertices n (labeled 0 up to $n - 1$) are in the graph and then a second line indicating how many edges m are in the graph. These two lines are followed by m lines indicating (bi-directional) edges as pairs of labels. For example,



is represented by

```
8
10
0 1
1 2
2 3
0 4
1 5
2 6
3 7
4 5
5 6
6 7
```

Immediately following the graph, an input line will indicate how many queries q there are, followed by q lines, each indicating a query as a pair of vertex labels.

Additional networks may follow, *not separated* by blank lines. Output is terminated with a line containing only a single “`0`”.

For each network, the output should begin with the line

`Network` r:

where r is the number of the network. For each of the corresponding queries $a\,b$, there should be a line

`All network monitors between` a `and` b: $\mathcal{L}$

where $\mathcal{L}$ is a nonempty list of monitors for the query or “`none`” if there are no monitors for the query.

Sample Input

```
3
2
0 1
1 2
1
2 0
8
10
0 1
1 2
2 3
0 4
1 5
2 6
3 7
4 5
5 6
6 7
2
3 0
1 7
0
```

Sample Output

```
Network 1:
All network monitors between 2 and 0: 1
Network 2:
All network monitors between 3 and 0: 1, 2
All network monitors between 1 and 7: none
```

PHONETIC MATCHING

You are playing the role of a newspaper reporter who wishes to build a program that will help flag press releases that come across the wire related to a political candidate that you are covering. Press releases come quite frequently from all of the candidates, so automating the process of identifying those from your candidate will reduce the amount of reading you need to do in order to keep on top of the race. Unfortunately, the candidate you are covering comes from a family that immigrated into the country from a place that didn't speak English, so the transliterated spelling of their name to English is non-unique from a phonetic point of view. Their name is easy to pronounce but difficult to spell correctly, so press releases frequently contain the name of the candidate spelled slightly incorrectly. Clearly an approach that searches on exact string matches will miss important press releases. Something more creative must be employed.

In researching the problem of identifying misspelled articles, you discover an algorithm from the early 1900s known as ”Soundex”. It is a basic algorithm in which words are given signatures based on their pronunciation. Words that are misspelled but are similar in pronunciation will be given identical signatures, allowing one to match based on signatures instead of exact string matching. You would like to build a tool that processes text files to identify press releases from your candidate using this algorithm. The algorithm for constructing the signature of a word is as follows:

1. Retain the first letter of the string.
2. Remove all occurrences of the following letters, unless it is the first letter: a, e, h, i, o, u, w, y.
3. Assign numbers to the remaining letters (after the first) as follows:
   - b, f, p, v = 1
   - c, g, j, k, q, s, x, z = 2
   - d, t = 3
   - l = 4
   - m, n = 5
   - r = 6
4. If two or more letters with the same number were adjacent in the original name (before step 1), or adjacent except for any intervening h and w then omit all but the first.
5. Return the first four characters, right-padding with zeroes if there are fewer than four.

As an example, both “Rupert” and “Robert” yield Soundex signatures “R163”, while Rubin produces “R150”.

The input text will have already been tokenized into a stream of words. The first line will contain two integers, the first being the number of candidates (k) and the second being the number of press releases (p). The next k lines will each contain the proper spelling of each candidate name. This will be followed by p blocks of words, each of which starts with the number of words (n) within the block, followed by n lines each containing a single word to process.

The output should be a table with p columns and k rows, with columns separated by a single space. For each document that contains a candidate name via a Soundex match, an `X` will be placed in the corresponding column of the candidate row. If there is no match, an `O` (oh, not zero) will be placed instead.

Sample Input

```
2 3
knuth
luks
4
knuth
likes
writing
algorithms
3
luks
grades
homework
5
knooth
and
lux
are
misspelled
```

Sample Output

```
X O X
O X X
```

SANE TRANSIT DISTRICT

Impressed with Lane County's goal of carbon neutrality, Kansas' Sane County is planning an extensive public transit system like LTD. The plan is for STD to use non-polluting street cars in the most effiicient way possible. Unlike busses, the street cars go from one end of the line to the other, then head "backward" along the same track. Since a street can accommodate at most one set tracks, it cannot be used by more than one streetcar.

The Commission has called for proposals to implement the street car system. However, it will only consider plans with the following features:

1. It must be possible to get from any stop to any other stop, possibly requiring some transfers from one streetcar to another.
2. The amount of track used must be the minimum amount required to connect all the stops.
3. For a given track/stop configuration satisfying the above conditions, the number of streetcars should be minimum.

Your program should take as input a set of street car lines and indicate whether modifications are needed to conform to Commission specifications

The flat Kansas terrain has enabled streets to be laid out in a perfectly square grid with blocks of length $\frac{1}{12}$ mile . The North-South streets are First Ave, Second Ave, etc and the East-West streets are First Street, Second Street, etc. Intersections are always desribed with the North-South street first. Thus, the corner of 42nd & 2nd, means 42nd Ave and 2nd St.

The first line input to this problem will contain an integer n indicating the number of proposals to follow. The first line of each proposal will then indicate the number s of streetcar lines in that proposal. The subsequent s lines will then indicate the route of a streetcar. Routes are listed via a comma-separated list of corner stops.[*] Each stop has the form i & j indicating the intersection of i^{th} Ave and j^{th} St.

Considering each proposal, you should determine whether constraints 1,2,3 are satisfied.

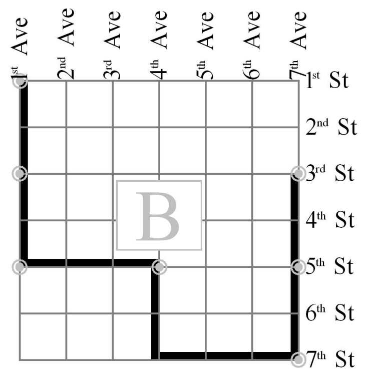
- If it is not possible to get from every stop to every other stop, your output should be "`Incomplete`".
- If condition 1 is satisfied, but the proposal is not using the minimum amount of track, your output should be "`Excess track`".
- Otherwise, your output should be "m `Streetcars`" where m is the minimum number of streetcars needed to traverse the track (and may be less than the proposed number).

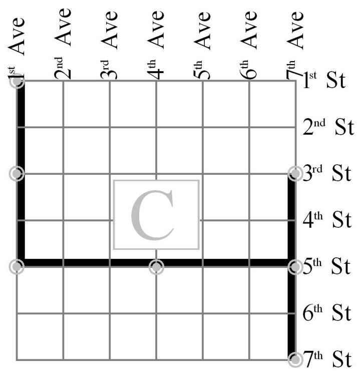
[*]Segments from stop to stop may involve turns with the exact route unspecified, but the length of the track is not affected by the choice of streets on which the streetcars make the turns.

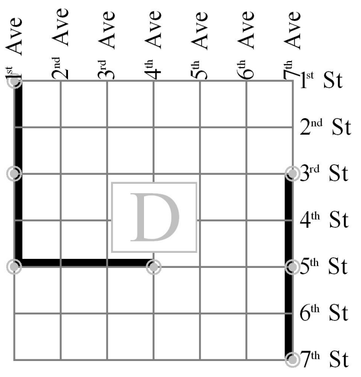
Sample Input

```
4
1&1,1&3,1&5,4&5,7&7,7&5,7&3
2
1&1,1&3,1&5,4&5,7&5
7&7,7&5,7&3
3
1&1,1&3,1&5
1&5,4&5,7&5
7&7,7&5,7&3
2
1&1,1&3,1&5,4&5
7&3,7&5,7&7
```

Sample Output	*Comments*
`Excess track`	see map A
`2 Streetcars`	see map B
`2 Streetcars`	see map B
`Incomplete`	see map C







Decimal Digit

All you need do in this problem is indicate the digit that lies at a given position in the decimal expansion of a given fraction.

The first line of the input will be an integer m indicating the number of problem instances to follow. Each of the subsequent m lines will contain a triple of integers $a\ b\ n$, with $1 \leq a < b < 10^6$ and $1 \leq n < 10^{15}$.

For each problem instance $a\ b\ n$, you should output the digit that lies in the n^{th} place in the decimal expansion of a/b.

Sample Input

```
3
1 2 4
3 7 2
4 9 1
```

Sample Output

```
0
2
4
```

Practice Program

This sample exercise asks you to write a program that adds and multiplies two integers.

Your program should expect input as follows: the first line will contain a single integer n followed by n lines, each of which has a pair of integers separated by a single space.

Each input integer pair should produce one output line that gives the sum and the product of the two integers separated by a single space. Following that should be a blank line and then your output should conclude with the name of your team followed by a blank line followed by the names of your team members, each on its own line.

Sample Input

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

Sample Output

```
4 0
6 5
5 6
usofa

North Carolina
South Dakota
West Virginia
```