DevCon-Xavier Code Challenge Cup (C-Cup)

C C++ C# Java JavaScript PHP Python Ruby VB.NET

November 30, 2012 • Xavier School Greenhills, San Juan City

Problem A: Best-Fit Lines

Mathematically, a line is defined in this particular form:

$$y = ax + b$$

where (x, y) are points of the line limited by the parameters a (slope) and b (y-intercept). A line equation is robust a model that it is normally used to help describe the behavior of certain variables. This is where the term "best-fit line" comes into the picture.

In order to determine the "best-fit line", the most common way is to find values of parameters a and b that can produce the lowest sum of the squares of "errors". An "error" is defined in this way: Given each actual point $(\hat{x}, \hat{y})$, the error is computed as $e = \hat{y} - (a\hat{x} + b)$.

Create a program that determines the "best-fit line" through the method described above.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename "bestfit.txt".

The first line of the file indicates the number of test cases (*n*), no more than 32767.

After the first line, there will be *n* sets of data, further defined as follows:

- The first line of the set indicates the number of points (*q*).
- The next *q* lines indicate the *x*-coordinates and *y*-coordinates of the points. They are encoded as two real numbers (with no more than two decimal points each), separated by a space. These are the actual points $(\hat{x}, \hat{y})$ described above.

Output Specifications

The program shall generate an ASCII file, with the filename "bestfit2.txt".

Within the file are *n* lines, each line indicating the values of a and b (with no more than two decimal points), separated by a space.

Sample Input

```
3
2
1 2
2 4
3
1.5 3.75
3 4.7
5.7 8
25
8 0
63 1
98 7
57 9
32 11
55 19
27 30
99 31
68 43
70 47
69 48
81 53
40 54
89 56
25 60
74 62
84 63
92 68
19 69
58 77
7 78
97 86
87 88
20 94
94 95
```

Sample Output

```
2 0
1.03 1.95
0.12 42.54
```

Problem B: Weird Writing

On the planet Ripley, a group of astronauts discovered some artifacts that would prove the existence of intelligent life outside of Earth. In fact, they were astounded because certain writings found in these artifacts looked like a typical Roman alphabet script (i.e. Earth's typical alphabet for English, like what is used in this problem set).

However, Ripley is already a desolate planet with no further evidence of life, so these astronauts were unable to verify these writings. Fortunately, though, they came up with strong conclusions about these writings, particularly of the existence of an order in their scripts (very much the same as how we Earthlings would alphabetize things).

The challenge they faced was that the Ripley alphabetization does not follow our own English alphabetization. For example, the following line below is how the English alphabet is ordered.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

On the other hand, the Ripley alphabet is ordered in the following manner.

```
Q W E R T Y U I O P L K J H G F D S A Z X C V B N M
```

(Oddly enough, it looks like it was lifted from a keyboard …)

In any case, these astronauts would like to you to sort a bunch of Ripley words with a particular Ripley alphabet.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename "ripley.txt".

The first line of the file indicates the number of test cases (*n*), no more than 32767.

After the first line, there will be *n* sets of data, further defined as follows:
- The first line of the set indicates the Ripley alphabet, with no spaces in between.
- The second line of the set indicates the number of words (*q*) to be sorted.
- The next *q* lines list down the words. Each line will only have one word. All letters will be in upper-case format with no other characters embed in between.

Output Specifications

The program shall generate an ASCII file, with the filename "ripley2.txt".

Within the file are *n* sets of words, sorted “Ripley-alphabetically”, with each set ending with “*” to indicate the end of the list for that set.

Sample Input

```
2
QWERTYUIOPLKJHGFDSAZXCVBNM
2
SCOTT
WEAVER
QWERTYUIOPLKJHGFDSAZXCVBNM
3
FUNNY
FACT
FEARSOME
```

Sample Output

```
WEAVER
SCOTT
*
FEARSOME
FUNNY
FACT
*
```

Problem C: Congressional Progression

In the republic of Freeport, a benevolent ruler named Leon Guerrero has reigned for quite a long time. Under his leadership, he has been able to maintain the economic growth of Freeport for the last 35 years. Considering that no one in Freeport has had any problems with his leadership, the republic of Freeport gave him free rein to decide on the orientation and direction of all legislative activities of Freeport. In short, all legislative powers belong to Leon Guerrero.

One day, Leon Guerrero decides to create a congress where representatives of each district shall be given full legislative autonomy. This was a breakthrough in the political dynamics of Freeport since this meant that their benevolent ruler shall relinquish all legislative authority to the congress after nearly a half-century rule, creating a more participative government. However, the problem lies in the determination of the districts in Freeport. To this, Leon Guerrero proposed one of the many principles.

```
Republic Act LG-23-45: An Act to Determine Major Districts in Freeport

WHEREAS, the republic of Freeport is fully connected with each town able to reach
other towns;
WHEREAS, there are certain towns that, if removed from Freeport, will cause
geopolitical divisiveness in Freeport;
WHEREAS, Republic Act LG-23-43 states the need to define districts for the congress
of Freeport;

I, Leon GUERRERO, by the power vested in me, enacts Republic Act LG-23-45 defining
the major districts in Freeport.

Article 1. Definition. A major district is a town that is considered as critical in
the connections across Freeport. If that town is removed from the map, there will
be certain towns from the remaining towns that will be disconnected from each other
as a result. As such, that town shall be considered as a major district.

…
```

As his lead staff in information technology, you are tasked to create a program to determine the major districts in Freeport, given adjacent connections of towns.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename “leon.txt”.

The first line of the file indicates the number of test cases (*n*), no more than 32767.

After the first line, there will be *n* sets of data, further defined as follows:

- The first line of the set indicates the number of connections (*q*).
- The next *q* lines list down the names of two towns, separated by a space. This means that both towns are immediate connected in the map of Freeport. Every town in Freeport only has one word for their name.

Output Specifications

The program shall generate an ASCII file, with the filename "leon2.txt".

Within the file are *n* sets of towns that are defined as major districts, with each set ending with "`*`" to indicate the end of the list for that set. In the event of no major districts, simply indicate "`None`".

Sample Input

```
3
2
Abalone Borgia
Camelville Borgia
3
Abalone Borgia
Camelville Borgia
Camelville Abalone
7
Camelville Borgia
Deloitte Camelville
Abalone Borgia
Evelyn Frances
Frances Deloitte
Abalone Camelville
Evelyn Deloitte
```

Sample Output

```
Borgia
*
None
*
Camelville
Deloitte
```

Problem D: Islands

The scientists from Institute of Geodesy and Cartography made a satellite picture of some part of the Pacific Ocean, which would be composed a map of islands.

Island is a set of square cells of land, where each cell is reachable from another by land in any of the eight (8) possible directions. Each island is surrounded by water. No two islands contact by their corners. There are no lakes inside an island.

Unfortunately, the rendered map became very cumbersome and they would require some outside help to do some island inventory.

This is where you come in.

You are to create a program that will count the number of islands with a given map plot.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename “island.txt”.

In the first line, there are two numbers *N* and *M* — the height and the width of a map ($1 \le N, M \le 5000$) – separated by a single space. Each of the next *N* lines contains *M* symbols describing the map: "~" stands for water, "*" stands for land.

Output Specifications

The program shall generate an ASCII file, with the filename “island2.txt”.

It shall be composed of one line, indicating the total number of islands in the given map.

Sample Input

```
7 6
~~~~~~
~*~**~
~***~~
~~~~~~
~*~~*~
~**~*~
~~~~~~
```

Sample Output

```
3
```

Problem E: Basketball

In basketball, a player's statistics like points, rebounds, and assists are recorded for analysis. Since the a typical basketball game is fast paced, they normally record what a player does (e.g. rebound the ball, make an assist, miss a free throw) by simply listing each action in a single statline per player. At the end of the game, each player's performance is recorded by simply processing the player's statline.

For example, a player's statline may look like this:

```
2XYRTXRA3YX012ARY
```

The player's statistics can then be determined using the following legend:

1 - made a 1 point shot or free throw
0 - missed a 1 point shot or missed free throw
2 - made a 2 point shot
3 - made a 3 point shot
X - missed a 2 point shot
Y - missed a 3 point shot
A - made an assist
R - made a rebound
T - committed a turnover

In our example, based on the player's statline, the player made one and missed one free throw, made two and missed three 2-pt shots, made one and missed three 3-pt shots, made two assists, caught three rebounds, and committed one turnover. His statistics for the game is typically shown as:

```
Points 8
     Freethrows 50% (1/2)
     2-point shots 40% (2/5)
     3-point shots 25% (1/4)
Assists 2
Rebounds 3
Turnovers 1
```

You must create a program that will read the players' statline from a text file and churn out statistics.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename “basket.txt”.

The input file will contain one or more lines. Each line will contain the player’s jersey number followed by a single space and then the player’s statline, as described above.

Output Specifications

The program shall generate an ASCII file, with the filename “basket2.txt”.

The output should be player statistics described above. The first line of each player’s statistics displays the jersey number of the player, preceded by the word “`PLAYER`”. The second line shows the number of points preceded by the word “`Points`”. The next three lines show the shooting percentages (preceded by “`Freethrows`”, “`2-point shots`”, and “`3-point shots`”), indented by five (5) spaces. The last three lines indicate the rest (“`Assists`”, “`Rebounds`”, “`Turnovers`”). Refer to the Sample Output section for a typical view of the whole thing.

In computing for shooting percentages, if no shots were taken by the player, then indicate 0%.

Sample Input

```
33 RR2XT11A22X01
32 22X200RAR2
23 X22RA1101A3Y
```

Sample Output

```
PLAYER 33
Points 9
     Freethrows    3/4 = 75%
     2-point shots 3/5 = 60%
     3-point shots 0/0 = 0%
Assists 1
Rebounds 2
Turnovers 1

PLAYER 32
Points 8
     Freethrows    0/2 = 0%
     2-point shots 4/5 = 80%
     3-point shots 0/0 = 0%
Assists 1
Rebounds 2
Turnovers 0

PLAYER 23
Points 10
     Freethrows    3/4 = 75%
     2-point shots 2/3 = 67%
     3-point shots 1/2 = 50%
Assists 2
Rebounds 1
Turnovers 0
```

DevCon-Xavier Code Challenge Cup (C-Cup)

C C++ C# Java JavaScript PHP Python Ruby VB.NET
November 30, 2012 • Xavier School Greenhills, San Juan City

Problem F: Utility Maximization

A typical consumer can be described as maximizing a particular utility function when deciding to purchase a set of goods. A common formula is the product of the quantity of good *i* raised to a certain power (known as the *substitutability scale* of each good).

$$U(x_1, x_2, x_3, x_4, \ldots, x_N) = x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3} x_4^{\alpha_4} \cdots x_N^{\alpha_N} = \prod_{i=1}^{N} x_i^{\alpha_i}$$

However, a typical consumer is always limited by one's income m and the price of the each good p_i being sold in the market. As such, no consumer can purchase a set of goods beyond what is allowed by one's budget.

You are tasked to help predict the optimal set of goods, given the substitutability scale of each good, the consumer's income and the price of each good.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename "utility.txt".

The first line of the file indicates the number of test cases (q), no more than 32767.

After the first line, there will be n sets of data, further defined as follows:

- The first line of the set indicates the number of goods (N).
- The next N lines list down the substitutability scale of each good and its market price, separated by a single space. The first pair shall indicate the substitutability scale of good 1, the second good 2, and so on.
- After N lines, the consumer's income (m) is indicated.

Output Specifications

The program shall generate an ASCII file, with the filename "utility2.txt".

Within the file are q sets of optimal quantities of each good, followed by "*".

Sample Input

```
2
2
0.5 10
0.5 5
100
5
2 4
4 10
2 2
1 100
3 4
3000
```

Sample Output

```
5
10
*
125
100
250
2.5
187.5
*
```

Problem G: Evolution of Industries

A general theory of evolution of economies hinges on a reality that products and factors of production are improved over time. Certain products or skills that a country has fully mastered usually become their main exports and this mastery is at the very heart of the theory of comparative advantage.

A group of physicists back in 2007 developed a concept known as a *product space*, a network of relatedness between products or their respective industries. It is claimed that countries can develop goods that are close to the ones they usually produce. By understanding how researchers and analysts form and understand product spaces, it is possible to understand how countries can evolve over time.

While these physicists were able to present and structure the concept of a product space, knowing how a country should move from one industry to another is unclear. The resulting product space was simply used to describe how various countries fared against each other, but it was not used to determine which industries need to be traversed in order to move from one industry to another with the highest rate of success.

You are tasked to create a program that determines the most likely trajectory of evolution between two industries.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename “export.txt”.

The first line of the file indicates the labels of the two industries (*START*, *END*) whose evolution trajectory is considered, separated by a space.

After the first line, there will be more lines describing the proximity of the two given industries in the country, further defined as follows: *LABEL1 LABEL2 PROB*

- *LABEL1* is the label of the first industry.
- *LABEL2* is the label of the second industry.
- *PROB* is a number between 0 and 1 that described the proximity or similarity of both *LABEL1* and *LABEL2*. This also described the likelihood that *LABEL1* evolves into *LABEL2*.
- There is a single space between *LABEL1* and *LABEL2*. This is also another single space between *LABEL2* and *PROB*.

Output Specifications

The program shall generate an ASCII file, with the filename “export2.txt”. Overall, it should show the most likely evolution trajectory from *START* to *END*.

Each line shall indicate a point (which is the label of an industry) in the evolution trajectory.

Sample Input	Sample Output

Sample Input

```
A F
A B 0.7
A C 0.2
F D 0.3
D E 0.4
D C 0.5
C B 0.5
E F 0.7
```

Sample Output

```
A
B
C
D
F
```

Problem H: Relatively Prime

The Euler function of a natural number *n* counts all positive numbers less than *n* that is relatively prime to *n*. In other words, the greatest common factor of all such positive numbers with respect to *n* is 1.

You are tasked to create a program that will compute for the Euler function of a particular number.

Input Specifications

A simple flat ASCII file shall be source of all input data, with the filename "euler.txt".

Each line is a positive number *n*, which is no more than 65535.

Output Specifications

The program shall generate an ASCII file, with the filename "euler2.txt".

Each line shall return the value of its respective Euler function.

Sample Input	Sample Output
3	2
7	6
100	40
1024	512

Problem I: Jugs

You have two jugs, A and B, and an infinite supply of water. There are three types of actions that you can use: (1) you can fill a jug, (2) you can empty a jug, and (3) you can pour from one jug to the other. Pouring from one jug to the other stops when the first jug is empty or the second jug is full, whichever comes first. For example, if A has 5 gallons and B has 6 gallons and a capacity of 8, then pouring from A to B leaves B full and 3 gallons in A.

A problem is given by a triple (Ca,Cb,N), where Ca and Cb are the capacities of the jugs A and B, respectively, and N is the goal. A solution is a sequence of steps that leaves exactly N gallons in jug B. The possible steps are

```
fill A
fill B
empty A
empty B
pour A B
pour B A
success
```

where "`pour A B`" means "pour the contents of jug A into jug B", and "`success`" means that the goal has been accomplished.

You may assume that the input you are given does have a solution.

Input Specifications

The input file has the filename "jugs.txt". Input to your program consists of a series of input lines each defining one puzzle. Input for each puzzle is a single line of three positive integers: Ca, Cb, and N. Ca and Cb are the capacities of jugs A and B, and N is the goal. You can assume 0 < Ca <= Cb and N <= Cb <=1000 and that A and B are relatively prime to one another.

Output Specifications

The output file has the filename "jugs2.txt". Output from your program will consist of a series of instructions from the list of the potential output lines which will result in either of the jugs containing exactly N gallons of water. The last line of output for each puzzle should be the line "success". Output lines start in column 1 and there should be no empty lines nor any trailing spaces.

Sample Input

```
3 5 4
5 7 3
```

Sample Output

```
fill B
pour B A
empty A
pour B A
fill B
pour B A
success
fill A
pour A B
fill A
pour A B
empty B
pour A B
success
```

Problem J: Just the Facts

The expression N!, read as "N factorial," denotes the product of the first N positive integers, where N is nonnegative. So, for example,

```
 N        N!
 0         1
 1         1
 2         2
 3         6
 4        24
 5       120
10   3628800
```

For this problem, you are to write a program that can compute the last non-zero digit of any factorial for (0 <= N <= 10000). For example, if your program is asked to compute the last nonzero digit of 5!, your program should produce "2" because 5! = 120, and 2 is the last nonzero digit of 120.

Input Specifications

The input file shall have a filename of "facts.txt". Input to the program is a series of nonnegative integers not exceeding 10000, each on its own line with no other letters, digits or spaces. For each integer N, you should read the value and compute the last nonzero digit of N!.

Output Specifications

The output file shall have a filename of "facts2.txt". For each integer input, the program should print exactly one line of output. Each line of output should contain the value N, right-justified in columns 1 through 5 with leading blanks, not leading zeroes. Columns 6 - 9 must contain " -> " (space hyphen greater space). Column 10 must contain the single last non-zero digit of N!.

Sample Input

```
1
2
26
125
3125
9999
```

Sample Output

```
   1 -> 1
   2 -> 2
  26 -> 4
 125 -> 8
3125 -> 2
9999 -> 8
```

Community and Media Partners