2012 UTS Programming Competition

Saturday 11 August, 2012

Sponsored by:
The School of Computing and Communications, Faculty of Engineering & IT, and Google
Problem 1: ASCII triangles

Problem Description
In the days before the World Wide Web, computer nerds would communicate on BBSs over 300 baud modems. It would take hours to download a single GIF image, so they resorted to ASCII art. Let's draw some simple ASCII triangles to commemorate those pioneers.

Problem Task
Write a program that reads an integer N from standard input, and prints out a triangle of asterisks, made of N lines. (1<=N<=80)
The first line has one asterisk and the number of asterisks in each subsequent line increments by one. There is no whitespace before, between, or after the asterisks.

<table>
<thead>
<tr>
<th>Sample input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>**</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>****</td>
</tr>
<tr>
<td></td>
<td>*****</td>
</tr>
</tbody>
</table>
Problem 2: The Great Joy of Harshad Numbers

Problem Description:
A Harshad number is an integer that is divisible by the sum of its digits. The name Harshad means “giver of great joy” in Sanskrit, which makes them an Indian cousin of Happy numbers. Write a program that determines whether a given integer is a Harshad number or not.

(All calculations are in base 10).

Problem Task:
Each line of input is a single number to be tested, N. (1<=N<=9999999)
There may be any number of lines of input.

Your output for each number tested is a single line, starting with the number, followed by a space, followed by a character string signifying the result. If the number is a Harshad number, the result string is “GREAT JOY”. If not, print “sadness”.

Remember, judging of your program's output is case sensitive.

Sample input
6
101
42
999
1138

Sample output
6 GREAT JOY
101 sadness
42 GREAT JOY
999 GREAT JOY
1138 sadness
Problem 3: Where's Molly?

Problem Description:
In 1987, a revolutionary new children's book was published. The title was “Where's Molly?” and it was a puzzle book. Every page was filled with apparently random text, but at one location on each page the word “Molly” was hidden in the text. The reader had to find “Molly” as quickly as possible. The “Where's Molly?” craze has grown to include an international competition where contestants race to be the first to find all the Mollys in a new edition of the book. The winner gets one million dollars and a rare golden ticket to tour the Molly Minka Chocolate Factory.

You are a member of the Possible Missions Force (PMF), a secret government agency that is investigating disturbing rumors that Molly Minka is actually master criminal Ernie Blowhard in disguise. Your mission, should you choose to accept it, is to win the golden ticket at this year's International Where's Molly? Competition. Roland Hand, another member of your team, will enter the competition wearing glasses that contain a hidden camera, which will send an image of each page to you in the non-descript black van outside. You will run the image through optical character recognition software, and then through a special program designed to find the word “molly” on the page. You will then radio back to Roland's earpiece, telling him the line on which the word will be found. It is a foolproof plan, except for the fact that you haven't written the program yet.

As always, if caught, PMF will disavow any knowledge of your activities. Good luck! This message will not self-destruct.

Problem Task:
The first line of input is the number of pages, P. (1<=P<=100)
This is followed by the input data for P pages.

For each page:
The first line of input for each page is the number of lines on the page, N. (1<=N<=100)
The next N lines are the lines of text on the page. Each line is terminated by a newline character.
Each line consists of a mix of up to 100 lowercase letters and uppercase letters. Line numbers are not provided in the input.

Only one line on each page will contain the word “molly”, but it may be made up of any combination of upper and lower case letters, e.g. “molly”, “MOLLY”, “mOIlY”, “MoLLY” are all valid representations of “molly”.

Your output for each page is the line number on which the word “molly” appears. Count the lines starting from one, i.e. the first line is line 1. Output the answer for each page on a separate line.

Sample input
2
4
twasbrilligandtheslithytoves
didgryreandMOLLYnthewabedallmimsysweretheborogovesandthemomerathesoutgrabe
3
molyomlyollyollyWallygollybrollyMoLeXpOLyjoLLYoGerFOLLyGirlyMOyLLmOllYfUbAR

Sample output
2
3
Problem 4: Jacobite Number Calculator

Problem Description
The Jacobites were a Mediterranean people living in antiquity, whose homeland, Jacobia, was -- for a significant portion of its history -- under Roman rule. During this time, the Jacobites maintained their own distinct culture and traditions, as well as adopting and adapting elements of Roman culture. For example, the Jacobites adopted the Latin alphabet, since they lacked a writing system for their own language. However, when it came to transcribing numerical quantities, rather than adopting the Roman numeral system, the Jacobites developed their own unique system.

The most distinct feature of the Jacobite numeral system (as opposed to Roman numerals) is that all twenty-six letters of the Latin alphabet are utilised. The Jacobites allocated a numerical value to each letter, as follows:

<table>
<thead>
<tr>
<th>Ones</th>
<th>Tens</th>
<th>Hundreds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobite</td>
<td>Hindu-Arabic</td>
<td>Jacobite</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>J</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>K</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>L</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>M</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>N</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>O</td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>H</td>
<td>8</td>
<td>Q</td>
</tr>
<tr>
<td>I</td>
<td>9</td>
<td>R</td>
</tr>
</tbody>
</table>

There is no way to represent zero (0) using Jacobite numerals, nor can fractional (non-integer) amounts be represented, since the Jacobites used numbers for the sole purpose of counting.

Note also that 900 is not represented with its own distinct symbol. The Jacobites combined Z and S (or 800 + 100) in order to represent 900 as the digraph ZS. This is one of only a few permissible digraphs in Jacobite numerals (see below for more combinations). For example, one cannot combine Y (700) and T (200), nor even S and Z, to represent 900 -- only ZS is acceptable. Another example: UU (300 + 300) is not an acceptable representation of 600 -- only X is.

To represent numerical amounts between 1 and 999, the Jacobites adopted a place value system, akin to the standard decimal (Hindu-Arabic) system in use today. Written from left to right, "hundreds" are placed first, followed by "tens", then "ones". Examples: WOC = 563, KG = 27, F = 6, ZSR1 = 999.

However, there is a minor complication in the system, owing to the fact that the Jacobites were a God-fearing people. The Jacobites treated the name of their God, JHVH (whose pronunciation is unknown to modern scholars), with so much reverence that it was undesirable -- in fact, sacrilegious -- to inscribe the "Divine Name" on any material that might be idly discarded, especially for non-religious purposes. Accordingly, the letters JH and VH, which could be used to write the "Divine Name", were considered to be "forbidden" combinations in Jacobite numerals and therefore needed to replaced as follows:
- JH = 10 + 8 = 18, with II = 9 + 9 (which also happened to spell the Jacobite word for "life", which is why 18 was an auspicious number in Jacobite culture)
- VH = 400 + 8 = 408, with VDD = 400 + 4 + 4

These are the only instances in which consecutive repetition of numerals is permissible.

To represent amounts equal to 1000 or greater, the Jacobites used a symbol akin to an asterisk or bullet (*) to serve as both a separator and a multiplier. The symbol has the effect of multiplying the numbers preceding it by 1000, as well as adding the numbers that follow the symbol, if any. Its usage is best illustrated with a few examples:

- 1,000 = A*
- 2,340 = B*UM
- 18,408 = II*VDD
- 4,000,002 = D**B
- 39,218,995 = LI*TII*ZSRE
- 2,147,483,647 = B*SMG*VQC*XMG

Problem Task
Write a program that takes two numbers, written in Jacobite numeral form as described in the previous section, adds them together and outputs the result in Jacobite numeral form.

All Jacobite numerals shall be in UPPER CASE and the only other permissible character is the asterisk: *

The two addends are to be written on a single line, separated by a single space, with no leading or trailing whitespace. As the input for the program shall originate from standard input, the program shall be able to handle multiple lines of input, perform calculations on each line and output the results on one line per input set.

The program shall only add two numbers and output a number if the two input numbers are in the correct format. Three hashes: ### shall be output to signify an input error if either or both of the input numbers are in an incorrect format, i.e. there are forbidden characters (lowercase alphabetic, other non-alphabetic other than *), forbidden combinations (JH, VH), or nonsensical combinations (e.g. XXX, ABC, *F)

The program shall only handle numbers within the range 1 ≤ n ≤ 2,000,000,000. If either of the input numbers or the output number exceeds 2,000,000,000, the program shall output an overflow error, to be represented using three question marks: ???

Sample input
NE C
A*XNA*ZSQRD G*KC*UKF
JE C
B*** A
UJH X
LG DDD

Sample output
NH
H*XPE*UI
II
###
###

NB: The Jacobite people and numeral system are a COMPLETE FABRICATION on the author's part, created for the purpose of providing an interesting context for the problem. Please do not consult encyclopaedias, history books, etc to learn more about the "Jacobites" as described herein, since you will not find anything, or, worse, create a fake entry on Wikipedia in an attempt to perpetuate the fabrication (this author has far too much integrity to do such a thing).
Problem 5: Save the Scruffulas

Problem Description:
You work for LRX Enterprises, a company that specialises in electrified fences. The Department of Conservation has hired LRX Enterprises to protect the endangered Scruffula Trees from extinction. As you know, Scruffula Trees are used to produce Thnonts. A Thnont’s a great something that everyone wants. The Twiceler Corporation has cut down nearly all the Scruffula Trees in the world to satisfy Thnont demand. Now there are only a few thousand trees left, scattered in small groups in the National Parks. Although it is illegal to cut down Scruffula Trees, Twiceler is sending people into the park under the cover of night to do so. Your company, LRX will supply electric fences to stop them.

LRX's electric fences are all perfect circles, as that is the most efficient energy design. However, the trees are located in irregular groups. National Parks Officers have surveyed the trees, and for each park, they have chosen a set of circles that will cover all of the Scruffula Trees. One problem remains to be solved: some of the circles overlap. If two electric fences intersect, they will short out, rendering them useless. You have designed a junction pole that will allow two fences to cross and remain operational. LRX will deploy a junction pole at every point where fences cross or touch. It is your job to calculate how many junction poles will be needed for each park.

Problem Task:
Given the data that describes all the circular fences needed in each park, calculate the number of intersections, and hence the number of junction poles required for each park.

The first line of input will be the number of National Parks.
The data for each park then follows.
For each park, the first line is the number of circles needed, C. 1<=C<=100.
There will then follow C lines, one for each circle, with three floating point numbers:
X Y R
where X and Y are cartesian coordinates of the centre of the circle, and R is the radius (0<R<=100.0). The values of X and Y are both between -100.0 and 100.0.

Your output for each park is a single number on a line by itself, reporting the total number of points of intersection for the circles in that park.

One tricky aspect of this problem is dealing with floating point error – if three circles intersect at exactly the same point, floating point error in your calculations may show three distinct points of intersection. Fortunately, your boss tells you that you don't need to worry about that – none of the intersection points are remotely close to any other. Each intersection point involves only two circles. Also, there are no circles that touch tangentially at one point. Provided your calculations are accurate to at least 3 decimal places, you can ignore floating point error.

Sample input
2
2
0 0 1
1 0 1
3
0 0 1
1 0 0.8
-1 -1 0.8

Sample output
2
4
Problem 6: Extended Fizzbuzz

Problem Description
Fizzbuzz is a concentration game (or a drinking game) in which players take turns to count upwards from 1, saying the next number out loud: 1, 2, 3, etc. All numbers that are divisible by 3 should be replaced by the word 'fizz'. All numbers that are divisible by 7 should be replaced by the word 'buzz'. If a number is divisible by both 3 and 7, it is to be replaced by 'fizzbuzz'.

Easy, yes?
Yes – it is much too easy for elite programming minds such as ours. We play Extended Fizzbuzz.

In Extended Fizzbuzz, Player 1 first defines \( i \) numbers that are to be replaced by words, by making \( i \) statements with the format: “Any number divisible by \( q_i \) must be replaced by the word \( \text{word}_i \)” (where \( q_i \) can be any positive integer, and \( \text{word}_i \) can be any word). If a number is divisible by more than one of the \( q_i \), the player must say each word, in order of ascending \( i \).

For example, if the first player defines the replacement rules by saying, “Any number divisible by 2 must be replaced by the word 'even', any number divisible by 3 must be replaced by the word 'triple', and any number divisible by 4 must be replaced by the word 'quad' ” then the number 12, which is divisible by all three, must be replaced with “eventriplequad”. “quadtripleeven” or “triplequadeven” are incorrect, as the words are in the wrong order.

After defining the \( q_i \) and \( \text{word}_i \), Player 1 starts the game by saying any positive integer that is not divisible by any of the \( q_i \). After that, play proceeds around the circle of players, with each player adding 1 to the previous number, and replacing it with the appropriate words.

Problem Task
Write a program that simulates a game of Extended Fizzbuzz.
The first line of input is a single positive integer denoting the number of input cases (each input case represents a new game). This is immediately followed by the input cases.

For each input case:
The first line of input is a single positive integer \( n \), denoting the number of players. \( n < 10 \). The second line of input is a single positive integer \( r \) (1 <= \( r \) <= 10), denoting the number of replacement rules.
The next \( r \) lines will be replacement rules with format \( q_i \) \ \( \text{word}_i \), where \( q_i \) >=2, and \( \text{word}_i \) is a string of up to 10 lower-case letters. They are separated by a space. The next and final line for each case is a pair of numbers separated by whitespace \( a \ \ \ b \)
denoting the starting and ending numbers for the game (inclusive). 0 < \( a \) < \( b \) < 1000.

The output of the program will be the correct utterances of each player in turn, starting with “Player 1: \( a \)”, and ending with the player whose utterance corresponds to \( b \). Each player's turn should be printed on a new line. Note that there is a space between “Player” and the player number; no space between the player number and the colon; and a space between the colon and the player's utterance. When multiple replacement rules are used, the words are concatenated without spaces. An empty line must be printed between cases.
The sample input for Extended Fizzbuzz below contains 2 cases. Case 1 has 3 players, Case 2 has 2 players. The replacement rules for the first case are the same as for standard Fizzbuzz.

<table>
<thead>
<tr>
<th>Sample input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Player 1: 1</td>
</tr>
<tr>
<td>3</td>
<td>Player 2: 2</td>
</tr>
<tr>
<td>2</td>
<td>Player 3: fizz</td>
</tr>
<tr>
<td>3 fizz</td>
<td>Player 1: 4</td>
</tr>
<tr>
<td>7 buzz</td>
<td>Player 2: 5</td>
</tr>
<tr>
<td>1 10</td>
<td>Player 3: fizz</td>
</tr>
<tr>
<td>2</td>
<td>Player 1: buzz</td>
</tr>
<tr>
<td>3</td>
<td>Player 2: 8</td>
</tr>
<tr>
<td>2 even</td>
<td>Player 3: fizz</td>
</tr>
<tr>
<td>3 triple</td>
<td>Player 1: 10</td>
</tr>
<tr>
<td>4 quad</td>
<td>Player 1: 11</td>
</tr>
<tr>
<td>11 19</td>
<td>Player 2: eventriplequad</td>
</tr>
<tr>
<td></td>
<td>Player 1: 13</td>
</tr>
<tr>
<td></td>
<td>Player 2: even</td>
</tr>
<tr>
<td></td>
<td>Player 1: triple</td>
</tr>
<tr>
<td></td>
<td>Player 2: evenquad</td>
</tr>
<tr>
<td></td>
<td>Player 1: 17</td>
</tr>
<tr>
<td></td>
<td>Player 2: eventriple</td>
</tr>
<tr>
<td></td>
<td>Player 1: 19</td>
</tr>
</tbody>
</table>
Problem 7: Disk Scheduling

Problem Description:
A common problem faced by operating system designers is that of hard disk scheduling. Various applications running on the system will request to read from or write to a particular sector on the disk. It takes time to move the disk's read/write head from one sector to another, and most of this time is seek time – the time it takes to move the disk arm from one cylinder to another.

Requests for disk access may come in faster than the disk arm can move to accommodate them. In this case, they are put in a queue, and dealt with one at a time. There are several ways an operating system can choose which request to service next from the queue. Two algorithms are:

First Come First Served (FCFS): the requests are serviced in the order that they arrived.
Shortest Seek Time First (SSTF): selects the request in the queue that has the smallest seek time from the current read/write head position. In other words, it chooses the request whose cylinder is closest to the current cylinder.

For both of these algorithms, the disk arm may change direction for each request, or it may continue in the same direction.

Operating system designers want to choose the method that is the fastest, i.e. the method that services all requests in the queue in the least total time. This is approximately proportional to the total number of cylinders travelled by the read/write head. The number of cylinders travelled by the disk head between two requests is simply the absolute value of the difference between the two cylinder numbers of the requests.

Your job is to implement a disk scheduling simulator that will compare the performance of FCFS and SSTF for a given request queue.

Problem Task:
Each line of input represents one disk queue. There may be any number of lines of input.

The first number on each line is the starting position of the disk head. This is the cylinder of the disk request that has just been serviced. The remaining numbers on each line form the current disk queue. Each queue is a comma-separated list of cylinder numbers, with each cylinder number representing a single disk access request. Each cylinder number is an integer between 0 and 2047 inclusive. There may be up to 100 requests in the queue. The queue is ordered according to when the requests arrived, with the first (leftmost) being the earliest request.

Your output for each queue is two integers on a single line, separated by a single space. The first integer is the total number of cylinders travelled by the disk head if FCFS is used to service the queue. The second integer is the total number of cylinders travelled by the disk head if SSTF is used to service the queue.

Sample input
10,20,30
200,100,200,100,200,100
0,500,300,2047

Sample output
20 20
500 100
2447 2047
Problem 8: Two-legged taxi

Problem Description:
Rick Shaw is a rickshaw driver. Not a new-fangled motor rickshaw or cycle rickshaw – Rick's rickshaw is a cart which he pulls behind him as he runs on his own two feet. So Rick always takes the shortest route to his passenger's destination... except when he picks up passengers from the airport.

Many of the people Rick picks up at the airport are business travellers who want to go to a meeting in town via their hotel. Often the meeting place is a different hotel! This means Rick sometimes has to run much further than if he were to take his passenger directly to their meeting place.

Problem Task:
Rick has asked you to write an app for his mobile phone that will tell him the shortest route from A to B, via C. There are signposts at every junction, so as long as Rick knows the next junction to go to, he can choose the correct road to take.

First you will read in a list of up to 100 roads. Each road joins two junctions, without passing through any others. Also, every road is a two-way street. The first line of input is the number of roads in the network, R (1≤R≤100). The next R lines describe one road per line, in the format:
JunctionName1 JunctionName2  Distance
where JunctionName1 and JunctionName2 are strings of up to 20 lower case letters (no spaces). The airport, hotels and various meeting places are all located on junctions bearing their names. Distance is the length of the road in km. It is a floating point number between 0.0 and 100.0.

Next, you will read in a list of routes for which Rick needs to find the shortest distance. Each route is a list of three junctions, separated by spaces. The first junction will always be the airport. The second junction listed is the first stop, and the third junction listed is the final destination. There is no fixed number of routes – keep reading in routes until the input ends.

Your output for each route is a list of junctions from the airport to the final destination, separated by spaces. This route must include the second junction listed in the input. Print one line per route.

A unique shortest route is guaranteed to exist for all routes in the input.

Sample input
9
airport yha 10
airport grandhotel 6.2
airport greensquare 7.1
greensquare yha 5
greensquare grandhotel 1.9
greensquare conventioncentre 5.4
yha conventioncentre 2.3
conventioncentre operahouse 4.1
greensquare operahouse 8.2
airport yha operahouse
airport grandhotel conventioncentre

Sample output
airport yha conventioncentre operahouse
airport grandhotel greensquare conventioncentre