ECOO 2012 Programming Contest

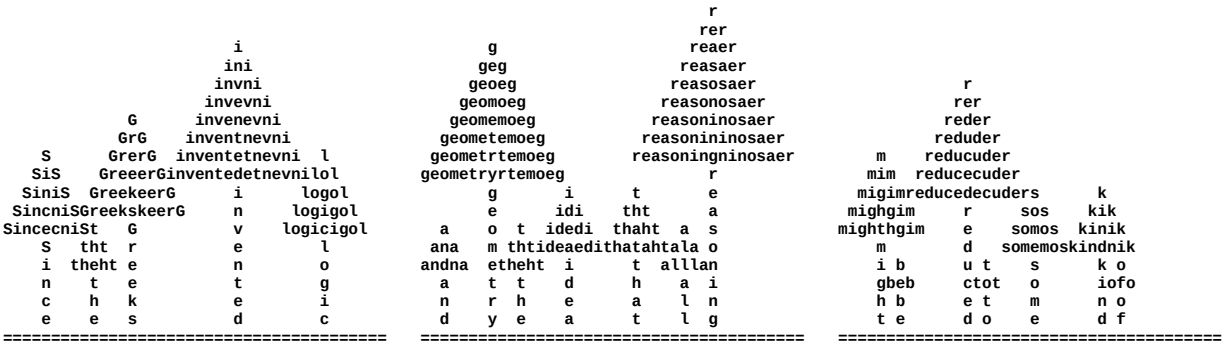
Final Competition (Round 3)

May 12, 2012

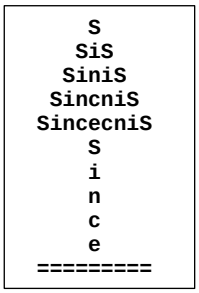
Problem 1: The Word Garden

In addition to creating problems for the ECOO competition, I have an unusual landscaping business on the side. I create custom-made “Word Gardens”. To make a Word Garden, I take one of my customers’ favorite quotations, turn each word in the quotation into a tree, and then plant the trees in rows, packing them as tightly together as I can.

Here’s an example, using a paraphrased quote from Hubert Dreyfus’ classic book, *What Computers Can’t Do*. The original quote was, “Since the Greeks invented logic and geometry the idea that all reasoning might be reduced to some kind of calculation has fascinated most of the West's rigorous thinkers”.



Notice that each word tree has a “trunk” consisting of the original word repeated twice vertically, and then “leaves” added to the top half of the trunk so that each level of leaves corresponds to an increasing portion of the word, repeated forwards and backwards and centered on the trunk. The tree for the word “Since” is shown at right.



The word trees are always planted in rows with an exact width of 40 characters. I plant the trees from left to right, placing each one as far to the left as I can without overlapping any previously planted tree. The ground consists of 40 “=” characters, and the maximum height of any tree is 23 characters (not including the ground).

DATA11.txt (DATA12.txt for the second try) will contain 5 lines of text. Each line of text will contain a list of words with each pair of words separated by a single space character. Each word will consist of a sequence of alphanumeric characters (and possibly also punctuation characters in some cases).

Each line of text should be turned into a line of trees, as described above. The ground should always have exactly 40 "=" characters. Each line of trees will always fit into a space of 24 rows by 40 columns. You should make sure you are using a fixed-width font (like courier) or the output will not look right.

Because of the size of the output, you are allowed to use keyboard input to show one row of trees at a time. You may also resize the output window and/or use the scroll bar to show the entire output during judging. If you need to, you may also copy and paste your output into Notepad or a similar text editor during judging.

Sample Input

```
Since the Greeks invented logic
and geometry the idea that all reasoning
might be reduced to some kind of
calculation has fascinated most of
the West's rigorous thinkers
```

Sample Output

See the previous page. Note that each new row of trees should appear in the output window *below* the row before, not beside it as shown here. The rows are only shown side by side to save space.

Problem 2: Jewelry Tips

There's a popular online game where the player gets an 8x8 board of coloured jewels and the goal is to form horizontal or vertical lines of 3 or more jewels of the same colour. This is done by repeatedly swapping pairs of adjacent jewels, either horizontally or vertically. A swap is only allowed if it would create at least 1 line of at least 3 jewels of the same colour. There are 7 colours: Red, Orange, Yellow, Green, Blue, Purple, and White.

Sometimes the players get stuck and have to be given a tip on what to do next. The tips fall into 3 categories: Normal, Good, and Excellent. A tip is "Normal" if acting on it would create a single line of 3 same-coloured jewels in a row. A tip is "Good" if acting on it would create a single line of 4 same-coloured jewels in a row. A tip is "Excellent" if acting on it would create either more than one line or a single line of 5. Here are some example boards. Underneath each board is the tip the game would give in that case.

```
BWPROYY
GBRBGWO
BPOOPPP
BWGYGWB
YGBYOBG
RGPOGOO
YBBYYOG
PBRYGPP
```

Norm: B.DN@0,0

```
YWPRRWGR
WGGPWOWP
PGWYOWO
YBWRGWP
WRGOYGP
GYPRWBB
BGWOPOB
ORRBBYP
```

Good: W.RT@1,4

```
GBWYPPOR
PPGWBYGP
OBORRWPR
RPGGOPOW
WBYYBBRP
GWPOWRGO
RPYOWPGO
RBWPGBOG
```

Excl: O.RT@7,6

Each tip starts with a four character coding of its type (`Norm`, `Good`, or `Excl`), then a colon and a space, then an 8 character code that gives the colour of the jewel to be swapped, the direction to swap it, and the location of the jewel before the swap. So "`Good: W.RT@1,4`" means "A good move is to take the white jewel at row 1, column 4 and swap it with the jewel to the right". The directions (up, down, left, and right) should be given using the two-character codes `UP`, `DN`, `LT`, and `RT`.

The colour of the jewel named in the tip must always match the colour of one of the lines that will be formed. For example, the tip in the first example could have been given as "`Norm: G.UP@1,0`", but the system would never express the tip this way because the line that would be formed if the player used the tip would be Blue, not Green.

Note that in the above examples, the first tip is "Normal" because it creates a single line of 3 blue jewels and no other lines, the second is "Good" because it will create a single line of 4 white jewels and no other lines, and the third is "Excellent" because it will create two lines of 3 jewels – one green and one orange.

The game's tipping system follows a few basic rules to decide which single tip to give, out of all the possible tips on the board.

Rules for Tipping

1. Don't give a good or excellent tip if there is a normal tip available.
2. Don't give an excellent tip if there is a good tip available.
3. When choosing between two tips of the same type on different rows, always choose the one from the highest row.
4. When choosing between two tips of the same type on the same row, always choose the one from the leftmost column.
5. In a situation where there is more than one possible tip of the same type for the same jewel, give priority to LT tips, then UP, RT and DN in that order.

In the first example above, other possible tips include "`Good: O.DN@4,4`" and "`Norm: O.RT@5,3`", but the first is prohibited by rule 1 and the second is prohibited by rule 3. In the second example, another possible tip would be "`Excl: W.DN@0,5`", but this is prohibited by rule 2. And in the third example, another possible tip would be "`Excl: G.LT@7,7`" but this is prohibited by rule 4.

DATA21.txt (DATA22.txt for the second try) will contain 10 test cases. Each test case is an 8x8 game board as shown below (note that the sample input shown only has 5 test cases). None of these game boards will contain a row of 3 or more jewels of the same colour. Your task is to write a program that will output a tip for each board, according to the specifications outlined above. If there are no tips to give, the program should simply output "Game Over".

Sample Input

```
YYBORBWG  GWOGWOGO  BOBWOPRP  OWBGYOYG  GGRBYWBB
RBPYRORY  PWOPGWRY  YORPBGOW  GGOYBGPW  RYWWOYWY
WYGBYPBO  PYBPBRRW  WWGYWWGB  WBPGRGWW  YYOBOPOP
BBRPBGPW  BOWGBBOG  RBRBWBOW  OPPWWRYP  WGBPBBYP
PWPYROBP  GWROGPWG  YWWGGRWR  YBORGOOP  RWRBYWGW
PYYRYWWR  PYPWBRRY  OGWRGBGB  WWOOWWRG  BROBOYRB
GPBYRYYG  POGRGPOO  BBPPWWOP  PPYBYPOG  OWGWPROP
PPWGOGPB  RYYOGBPW  WWPBORPB  RGOYPOWW  RWGWPGOO
```

Sample Output

```
Norm: P.RT@5,0
Excl: G.DN@4,4
Game Over
Norm: O.UP@7,2
Good: B.LT@3,4
```

Problem 3: Steam Arithmetic

Steam is a brand new dialect of Lisp, one of the oldest high-level programming languages. The name Lisp comes from the phrase "List Processing" because the only structure in Lisp (and therefore in Steam) is the list. A list is a sequence of tokens separated by whitespace and enclosed in round brackets. Here is a list with 6 elements: `(6 z 4.2 hello 4 world)`.

Lists can be nested, meaning they can have other lists as elements. Here is an example of a nested list: `(5 (x t e d) (6 7 (-1 2)))`. This list has three elements. The first is `5`, the second is the list `(x t e d)`, and the third is the list `(6 7 (-1 2))`, which also has another list nested inside of it. There is no limit to how many lists can be nested inside other lists.

Arithmetic in Steam uses "prefix" notation, in which the operator is written before the operands. We are used to seeing arithmetic in "infix" notation like this: `5 + 3`. But in prefix notation `"5 + 3"` becomes `"+ 5 3"`. In Steam, arithmetic expressions are lists like this:

`(+ 5 3)`	means	`5 + 3`
`(* 7 2)`	means	`7 * 2`
`(* (- 5 2) (+ 2 3))`	means	`(5 - 2) * (2 + 3)`

In Steam arithmetic, there are 5 operators: +, -, *, q (for quotient) and r (for remainder, which is the same as the standard "modulus" or "mod" operator for integers). Each operator takes exactly two integer operands. Every element in a list except for the last one is followed by a single space, and no other spaces are allowed anywhere in the list.

DATA31.txt (DATA32.txt for the second try) will contain 10 syntactically correct arithmetic expressions written in Steam, one on each line. To keep things simpler, only the integers 0 through 9 will be used as operands. Your job is to evaluate each expression and output the result. The result could be any positive or negative integer in the range $-32{,}768$ to $32{,}767$.

Sample Input

```
(+ 5 3)
(* 7 2)
(- 0 9)
(q 8 5)
(r 8 5)
(* (- 5 2) (+ 2 3))
(r (q 9 4) (* (+ (q 2 3) (- 5 (- 4 (* 2 3)))) 2))
(* 4 (* 4 (* 4 (* 4 (* 4 (* 4 4))))))
(* (* (* (* (* (* 4 4) 4) 4) 4) 4) 4)
(+ (q (- 9 3) (r 7 4)) (- (* 9 9) (* 0 1)))
```

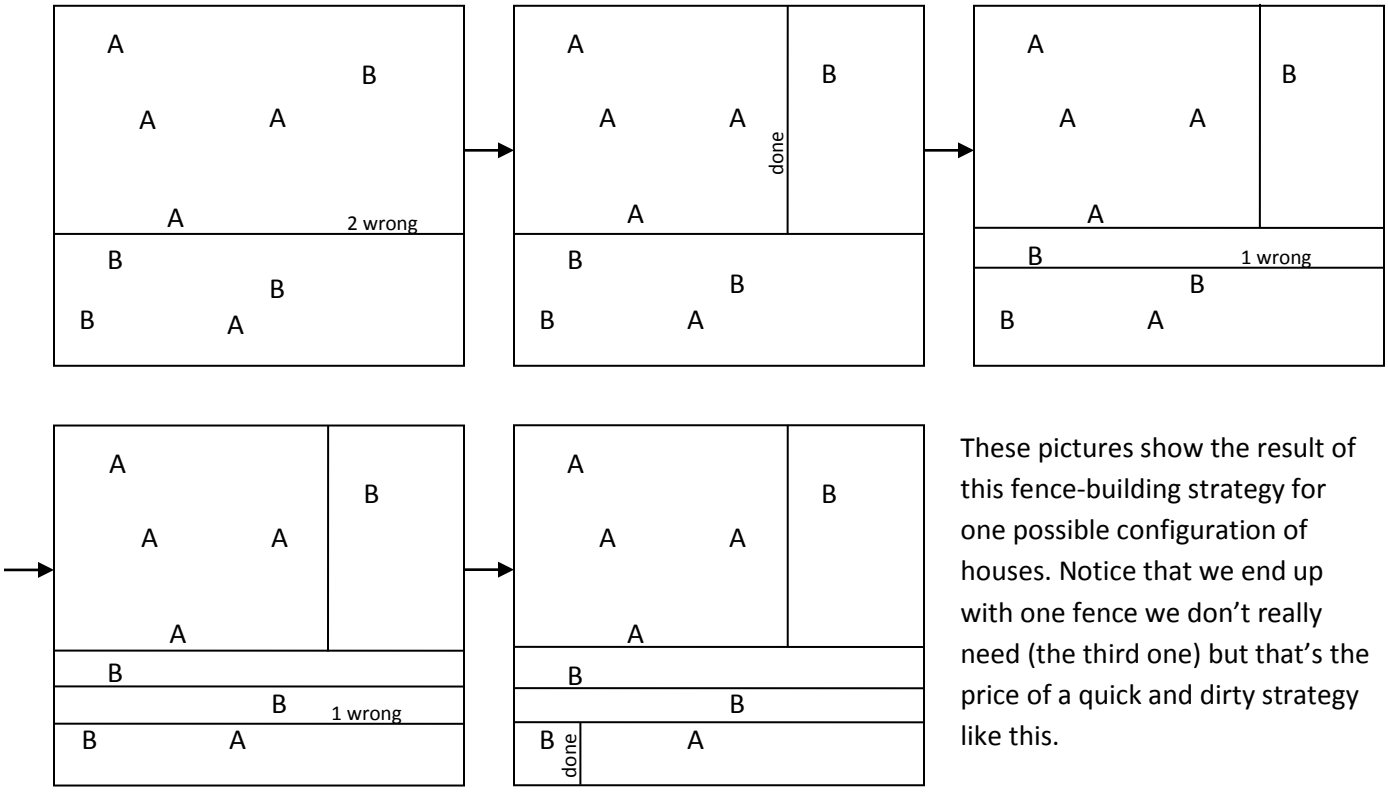
Sample Output

```
8
14
-9
1
3
15
2
16384
16384
83
```

Problem 4: Splitsville

A terrible feud has broken out in the French commune of Lacelle between families that use Android phones (A) and families that use Blackberries (B). Steps need to be taken immediately to keep the two groups apart. We have a map of where everyone lives and an unlimited budget to build electric fences. We don't have time to plan the best possible use of our fences, so we will go with a simple strategy that seems to get pretty good results most of the time.

1. We will only use fences running North to South (vertical) or East to West (horizontal).
2. When separating a region of the town, we will build the fence that best separates the houses in that region, creating two new regions. Then we will divide the new regions.
3. We will continue in this way until every region contains houses of only one family.
4. When deciding on the "best" fence in a region, we will go with the one that separates at least one house from the others and results in the fewest number of houses on the wrong side of the fence. (A house on the wrong side of the fence is a B house in a majority A area or an A house in a majority B area. If a region has the same number of A and B houses, then the A houses are considered to be on the wrong side.)
5. If there is a tie for the best fence between horizontal and vertical fences, we will use a horizontal one. If there is a tie among horizontal fences, we will use the northernmost one. If there is a tie among vertical fences, we will use the westernmost one.



These pictures show the result of this fence-building strategy for one possible configuration of houses. Notice that we end up with one fence we don't really need (the third one) but that's the price of a quick and dirty strategy like this.

DATA41.txt (DATA42.txt for the second try) will contain five test cases. Each test case consists of two lists of **X** and **Y** coordinates – one for the Android families, and one for the Blackberry families. Each list starts with the number of items in the list on a single line, followed by the **X** and **Y** locations of each house, one per line. The **X** coordinates are the East-West coordinates, with higher values being more Eastern locations. The **Y** coordinates are the North-South coordinates, with higher values being more Northern locations. Coordinates are integers ranging from -100 000 to +100 000.

Your output should consist of one number for each test case representing the number of fences you have to build in order to separate the families, following the strategy described above. The first test case below represents the example pictured above. There will always be at least 1 and at most 200 houses in the village. The maximum width and height of the village is 10 000 units. No two houses will ever share the same coordinates.

Sample Input

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

```
0 1
12 4
6 6
-7 -8
3 -5
10 10
10
-1 3
-1 4
-1 6
-1 8
-1 -100
-1 -7
-1 -15
-1 -17
-1 9
-1 -5
2
1 1
```

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

Sample Output

```
5
8
1
3
5
```