2018 APL Problem Solving Competition – Phase I Problem Descriptions

Phase I Tips

- The Phase I problems are designed to be solved using short APL dfns or tacit functions. If you find yourself writing more than a couple statements in your solution, you're probably doing it wrong.

- A dfn is one or more APL statements enclosed in braces {}. The left hand argument, if any, is represented in a dfn by ⍺, while the right hand argument is represented by ⍵.
  Example:

```
      'Hello' {⍺,'-',⍵,'!'} 'world'
Hello-world!
```

  A dfn terminates on the first statement that is not an assignment. If that statement produces a result, the dfn returns that value as its result. The diamond symbol ⋄ separates APL statements.
  Example:

```
      'left' { ⍵ ⋄ ⍺ } 'right'
right
```

  For more information on dfns, use the online help included with Dyalog APL or see page 152 in Mastering Dyalog APL (https://www.dyalog.com/uploads/documents/MasteringDyalogAPL.pdf).

- A tacit function is an APL expression that does not explicitly mention its arguments.
  In the example below (+⌿÷≢) is a tacit function which computes the average of a vector (list) of numbers.

```
      (+⌿÷≢) 1 2 3 4 5 6
3.5
```

- Each problem has a description and one or more examples. Wherever you see "***your_solution***" is where you should insert your solution (either a dfn or tacit function). You do not need to comment your solutions; the competition judging committee has decades of experience reading APL code.

- Your code must run in a default Dyalog environment using (⎕ML ⎕IO)←1. If you use other settings for ⎕ML or ⎕IO, they must be local. If you don't know what that means, don't worry, it won't matter to you.

- Several of the problem descriptions will describe arguments that can be scalar (a single value) or a vector (a list of values). This is largely pedantic, but in such cases your solutions should produce correct results for both scalar and vector cases.

- The symbol ⍝ is the APL comment symbol. In some of the example, we provide comments to give you more information about the problem.

  Some of the problem test cases use "boxed display" to make the structure of the returned results clearer. Boxing is enabled by default on www.TryAPL.org and can be enabled with the user command:

```
      ]box on
```

  Without boxed display enabled:

```
      ⍳¨⍳4
1   1 2   1 2 3   1 2 3 4
```

  With boxed display enabled:

```
      ⍳¨⍳4
┌─┬───┬─────┬───────┐
│1│1 2│1 2 3│1 2 3 4│
└─┴───┴─────┴───────┘
```

The following is a sample problem to show what a typical problem description looks like and presents some possible solutions of varying quality.

Sample Problem – Counting Vowels

Write a APL expression to count the number of vowels in a character vector or scalar.

Examples:

```
      your_solution 'APL Is Cool'
4

      your_solution ''    ⍝ empty argument
0

      your_solution 'N Vwls Hr'  ⍝ No Vowels Here
0
```

Below are 3 sample solutions. All 3 produce the correct answer, however the first 2 solutions would be ranked higher by the competition judging committee as they demonstrate better use of array-oriented programming.

"Good" Solutions

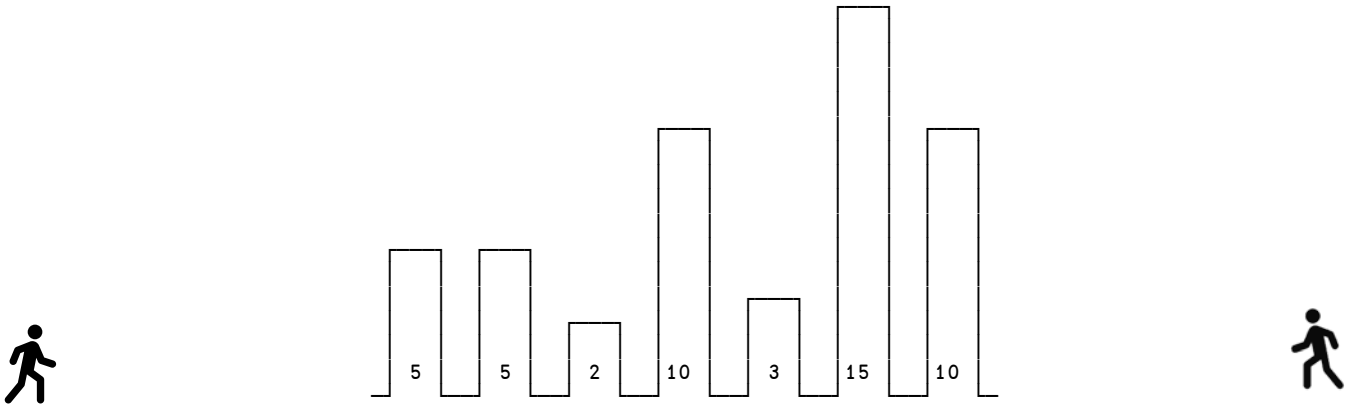
```
      {+/⍵∈'AEIOUaeiou'} 'APL Is Cool' ⍝ using a dfn
4

      (+/∈∘'AEIOUaeiou') 'APL Is Cool'   ⍝ using a tacit function
4
```

"Lesser (but still correct)" Solution

```
      {(+/⍵='A')+(+/⍵='E')+(+/⍵='I')+(+/⍵='O')+(+/⍵='U')+(+/⍵='a')+
(+/⍵='e')+(+/⍵='I')+(+/⍵='o')+(+/⍵='u')} 'APL Is Cool'
4
```

Problem 1 - Oh Say Can You See?

Imagine standing in front of a line of skyscrapers of varying heights. Assume that you can always see a skyscraper that is taller than a closer skyscraper. For example, the person depicted on the left can see 3 skyscrapers – the first, fourth, and sixth from left to right. Whereas the person on the right can see 2 skyscrapers – the seventh and sixth.



Write an APL expression that, given a scalar or vector of skyscraper heights from closest to furthest, will return an integer representing the number of skyscrapers that can be seen.

Examples:

```
      your_solution 5 5 2 10 3 15 10   ⍝ from the left person's view
3
      your_solution ⌽ 5 5 2 10 3 15 10 ⍝ from the right person's view
2
      your_solution ⍬   ⍝ no skyscrapers here!
0
      your_solution 10  ⍝ single skyscraper
1
```

Problem 2 - Number Splitting

Write an APL expression that, given a scalar real non-negative number, will return a two-element vector of the integer and fractional parts of the number.

Examples:

```
      your_solution 1.234
1 0.234
      your_solution 12
12 0
      your_solution .1234
0 0.1234
      your_solution 0
0 0
```

Problem 3 - Rolling Along

Using the key operator ⌸, write an APL expression that, given an integer scalar or vector representing the number of sides on each of a set of dice, will return a histogram showing the distribution curve for the possible totals that can be rolled using those dice. The histogram is a 2-column matrix where the left column contains the possible totals for the dice, and the right column has vectors containing asterisks representing the number of occurrences of the corresponding totals. Trailing spaces are allowed in the character vectors.

Examples:

```
      your_solution 6 6   ⍝ 2 6-sided dice
 2  *
 3  **
 4  ***
 5  ****
 6  *****
 7  ******
 8  *****
 9  ****
10  ***
11  **
12  *
```

```
      your_solution 6  ⍝ 1 6-sided die (flat distribution)
1  *
2  *
3  *
4  *
5  *
6  *
```

```
      your_solution 5 3 4  ⍝ 5, 3, and 4-sided dice
 3  *
 4  ***
 5  ******
 6  *********
 7  **********
 8  **********
 9  *********
10  ******
11  ***
12  *
```

```
      your_solution ⍬  ⍝ no dice
0  *
```

Problem 4 - What's Your Sign?

The Chinese animal zodiac is a repeating cycle of 12 years, with each year being represented by an animal. 2018 is the year of the dog. The signs for the last 12 years are:

2018	Dog	2012	Dragon
2017	Rooster	2011	Rabbit
2016	Monkey	2010	Tiger
2015	Goat	2009	Ox
2014	Horse	2008	Rat
2013	Snake	2007	Pig

Note that the year 1 AD (represented as `1`) follows the year 1 BC (represented as `¯1`) with no intervening "0" year.

Write an APL expression that, given a scalar integer year, returns a character vector (string) of the Chinese zodiac sign for that year. For the purposes of this problem, assume that each year number corresponds to exactly one Chinese zodiac animal.

Examples:

```
      your_solution 2018  ⍝ Newborns this year will be Dogs
Dog

      your_solution 1564  ⍝ William Shakespeare b. 1564
Rat

      your_solution ¯551  ⍝ Confucius b. 551 BC
Dog
```

Problem 5 - What's Your Sign? Revisited

In Western astrology, the Zodiac is based on twelve 30° sectors of the ecliptic. Although the exact dates in a given year may shift by a day, the general dates for each sign are:

Aries	March 21-April 19	Libra	September 23-October 22
Taurus	April 20-May 20	Scorpio	October 23-November 21
Gemini	May 21-June 20	Sagittarius	November 22-December 21
Cancer	June 21-July 22	Capricorn	December 22-January 19
Leo	July 23-August 22	Aquarius	January 20-February 18
Virgo	August 23-September 22	Pisces	February 19-March 20

Write an APL expression that given a 2-element integer vector representing month and day, returns a character vector (string) of the corresponding Western zodiac sign.

Examples:

```
      your_solution 2 23 ⍝ February 23
Pisces

      your_solution 10 31 ⍝ October 31
Scorpio
```

Problem 6 - What's Your Angle?

XML elements are denoted with content enclosed in beginning and ending tags. The tags themselves are enclosed in left and right angle brackets '<' and '>'. The only valid occurrences of angle brackets are as a part of beginning or ending tags.

For example, the following is valid
```
<name><first>Drake</first><last>Mallard</last></name>
```

Whereas the following is not valid XML
```
<math><relation>2<3</relation></math>
```

One easy validation is to check that the angle brackets are properly balanced – all left angle brackets '<' must be "closed" with right angle brackets '>' before another occurrence of a left angle bracket.

Write an APL expression that given a character scalar or vector representing some XML, returns 1 if the angle brackets are properly balanced and 0 if not.

Examples:

```
      your_solution '<name><first>Drake</first><last>Mallard</last></name>'
1

      your_solution '<math><relation>2<3</relation></math>'
0

      your_solution ''   ⍝ an empty vector is balanced
1

      your_solution '>stuff<>/stuff<'
0

      your_solution '<'
0
```

Problem 7 - Unconditionally Shifty

Logical bitwise shifting is a common operation where bits in a fixed width field are moved to the left or right by a specified amount, **N**, causing **N** bits to be "shifted out" at one end and **N** 0's to be filled in at the other end.

For example, shifting the following vector by 3

```
      1 0 1 1 1 0 1 1
```

would result in

```
      0 0 0 1 0 1 1 1
```

and shifting by ¯3 would result in

```
      1 1 0 1 1 0 0 0
```

Write an APL expression that given a right argument of a Boolean scalar or vector, and left argument scalar integer of the shift amount, returns an appropriately shifted transformation of the right argument. Solutions that do not use guards will be judged more favorably. For a description of guards, see page 215 in https://www.dyalog.com/uploads/documents/MasteringDyalogAPL.pdf.

Examples:

```
      3 your_solution 1 0 1 1 1 0 1 1
0 0 0 1 0 1 1 1

      ¯3 your_solution 1 0 1 1 1 0 1 1
1 1 0 1 1 0 0 0


      5 your_solution ⍬   ⍝ result is an empty vector



      0 your_solution 1 0 1 1 1 0 1 1
1 0 1 1 1 0 1 1

      3 your_solution 1
0
```

Problem 8 - Making a Good Argument

Part of Dyalog's help text for dyadic transpose **R←X⍉Y** states:

- **Y** may be any array.
- **X** must be a simple scalar or vector whose elements are included in the set **⍳⍴⍴Y**.
- Integer values in **X** may be repeated but all integers in the set **⍳⌈/X** must be included.
- The length of **X** must equal the rank of **Y**.

Write an APL expression that given a right argument (**Y**) of any array and a numeric scalar or vector left argument (**X**) returns a Boolean indicating if the left argument is a valid argument for **X⍉Y**.

Examples:

```
      3 1 2 your_solution 2 3 4⍴⍳24
1

      2 1 2 your_solution 2 3 4⍴⍳24
1

      2 3 2 your_solution 2 3 4⍴⍳24
0

      1 1 your_solution 3 4⍴⍳12
1

      1 2 your_solution 2 3 4⍴⍳24
0

      1.1 2 3 your_solution 2 3 4⍴⍳24
0

      1 your_solution ⍬
1

      ⍬ your_solution 1
1
```

Problem 9 – Earlier, Later, or the Same?

The system function ⎕TS returns a 7-element integer vector timestamp representing the current year, month, day, hour, minute, second, and millisecond in that order.

Write an APL expression that given left and right arguments of such timestamps returns a ¯1, 1, or 0 if the left argument represents a time that is respectively, earlier than, later than, or simultaneous with the right argument.

Examples:

```
      2018 4 1 12 34 56 789 your_solution 2018 4 1 16 45 12 800
¯1

      2018 4 1 12 34 56 789 your_solution 2018 4 1 12 34 56 789
0

      2018 4 1 12 34 56 789 your_solution 2017 4 1 12 34 56 789
1
```

Problem 10 – Anagrammatically Correct

An anagram is a word or phrase that can be formed by rearranging the letters of another. For instance, 'stained' and 'instead' are anagrams, as are 'emigrants' and 'streaming'. Spaces are not considered significant in the comparison.

Write an APL expression that takes left and right arguments of character scalars or vectors returns a **1** if the arguments are anagrams of one another, **0** otherwise. You may assume that both arguments are both either upper-case or lower-case.

Examples:

```
      'ALBERT EINSTEIN' your_solution 'TEN ELITE BRAINS'
1

      '' your_solution ''
1

      'd' your_solution 'd'
1

      'mesas' your_solution 'seam'
0

      'apple' your_solution 'lapel'
0
```