2017 International APL Problem Solving Competition – Phase I

Phase I Tips

- We have provided you with several test cases for each problem to help you validate your solution. In an APL session, user inputs are indented 6 spaces and we use that convention in our examples to help you distinguish between inputs and outputs.
- We recommend that you build your solution using dfns. 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.
  Example:
  ```
        'left' { ⍵ ⋄ ⍺ } 'right'
  right
  ```
  For more information on dfns, use the online help included with Dyalog or refer to *Mastering Dyalog APL* at http://www.dyalog.com/MasteringDyalogAPL/MasteringDyalogAPL.pdf.
- The symbol ⍝ is the APL comment symbol. In some of the examples below, comments are provided to give more information.
- 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 in your copy of Dyalog by entering:
  ```
        ]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

Sample Problem - I'd like to buy a vowel
Write a dfn to count the number of vowels in a character vector.

When passed the character vector 'APL Is Cool', your solution should return:
```
4
```

Below are 2 sample solutions. Both produce the correct answer, however the first solution would be ranked higher by the competition judging committee as it demonstrates better use of array oriented programming.

```
      {+/⍵∊'AEIOUaeiou'}'APL Is Cool' ⍝ better solution
4
      {(+/⍵='A')+(+/⍵='E')+(+/⍵='I')+(+/⍵='O')+(+/⍵='U')+(+/⍵='a')+
(+/⍵='e')+(+/⍵='I')+(+/⍵='o')+(+/⍵='u')}'APL Is Cool' ⍝ lesser solution
4
```

Problem 1 – What an Odd Bunch

Write a dfn that will return the first n odd numbers.

Test cases:

```
      {your_solution} 5
1 3 5 7 9

      {your_solution} 1
1

      {your_solution} 0   ⍝ should return an empty vector
```

Problem 2 – Good Evening

Write a dfn that takes an integer array and replaces all the odd numbers with the next greatest even number.

Test cases:

```
      {your_solution} 1 2 3 4 5
2 2 4 4 6

      {your_solution} ⍬ ⍝ should return an empty vector


      {your_solution} 4 4⍴⍳16 ⍝ should work with arrays of any rank
 2  2  4  4
 6  6  8  8
10 10 12 12
14 14 16 16
```

Problem 3 – Miss Quoted

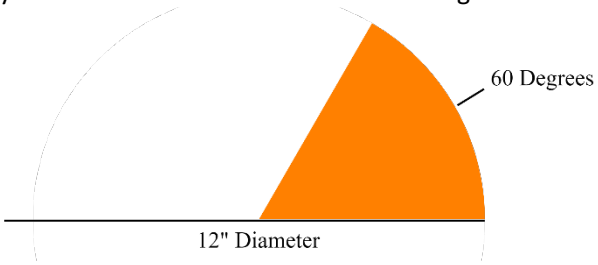
Write a dfn that will remove text found between pairs of double quotes (").
Hint: one technique is to use ≠\, but there are many ways to solve this problem.

Test cases:

```
      {your_solution} 'this "is" a test'
this "" a test

      {your_solution} 'this is a test'
this is a test

      {your_solution} 'this "is" a "test"'
this "" a ""

      {your_solution} ''  ⍝ should return an empty vector
```

Problem 4 – Slice(s) of Pie(s)

Write a dfn that calculates and returns the areas of 0 or more pie slices. The left argument is 0 or more angles (in degrees). The right argument is 0 or more pie diameters. If the number of angles and diameters are not equal to each other (and neither is a single number), a LENGTH ERROR should be generated.
Hint: If you use APL properly, you should not have to check for the length of either argument – it will just work.



Test cases:

```
      60 {your_solution} 12
18.84955592

      0 {your_solution} 12  ⍝ 0 degree slice
0

      60 {your_solution} 0  ⍝ 0 diameter pie
0

      60 {your_solution} 9 12 15  ⍝ 60 degree slices of 3 different pies
10.60287521 18.84955592 29.45243113

      60 90 120 {your_solution} 12 ⍝ 3 different size slices of the same pie
18.84955592 28.27433388 37.69911184

      60 90 120 {your_solution} 9 12 15 ⍝ different sizes of different pies
10.60287521 28.27433388 58.90486225

      60 90 120 {your_solution} 9 12 ⍝ 3 slices, 2 pies?
LENGTH ERROR
```

Problem 5 – DNA?

Write a a dfn that takes a string representing a nucleotide and returns a 1 if it is a valid DNA string, 0 otherwise. In other words, are all the characters in the string in the set `'ACGT'`?

```
      {your_solution} 'ATGCTTCAGAAAGGTCTTACG'
1

      {your_solution} 'Dyalog'
0

      {your_solution} ''       ⍝ an empty string is valid
1

      {your_solution} 'T'
1
```

Problem 6 – k-mers

The term k-mer typically refers to all the possible substrings of length k that are contained in a string. In computational genomics, k-mers refer to all the possible subsequences (of length k) from a read obtained through DNA Sequencing. Write a dfn that takes a character vector as its right argument and k (the substring length) as its left argument and returns a vector of the k-mers of the original string.

Test cases:

```
      4 {your_solution} 'ATCGAAGGTCGT'
```

ATCG	TCGA	CGAA	GAAG	AAGG	AGGT	GGTC	GTCG	TCGT

```
      4 {your_solution} 'AC'    ⍝  k>string length? Return an empty vector
```

Problem 7 – Counting DNA Nucleotides

Write a dfn that takes a DNA string and returns 4 integers of the number of occurrences for each of the symbols 'A', 'C', 'G', and 'T' respectively.

Test cases:

```
      {your_solution} 'AGCTTTTCATTCTGACTGCTGTCTTTAAAAAAAGAGTGTCTGATAGCAG'
14 8 10 17

      {your_solution} 'CCAAATGGGG
3 2 4 1

      {your_solution} ''
0 0 0 0

      {your_solution} 'G'
0 0 1 0
```

Problem 8 – Be the First 1

Write a dfn that takes a Boolean vector or scalar and "turns off" all the 1s after the first 1.

Test cases:

```
      {your_solution} 0 1 0 1 0 0 1
0 1 0 0 0 0 0

      {your_solution} ⍬ ⍝ should return an empty vector


      {your_solution} 0 0 0 0 ⍝ no 1's?  no problem!
0 0 0 0
```

Problem 9 – Double Trouble

Write a dfn that takes a character vector or scalar and returns a Boolean vector indicating anywhere an element is followed by an element of the same value.

Test cases:

```
      {your_solution} 'bookkeeper'
0 1 0 1 0 1 0 0 0 0

      {your_solution} ''  ⍝ should return an empty vector

      {your_solution} 'aaaaaa'
1 1 1 1 1 0

      {your_solution} 'd'
0
```

Problem 10 – Squaring Off

Write a dfn that will reshape a given array into the smallest square matrix that will contain all the elements of the argument, padding with additional elements if necessary. The pad element should be 0 if the array is numeric and space ' ' if the array is character.

Test cases:

```
      {your_solution} 1 2 3 4
1 2
3 4
      {your_solution} 1 2 3 4 5
1 2 3
4 5 0
0 0 0

      {your_solution} 'Dyalog APL' ⍝ should work with any data
Dyal
og A
PL


      ' '={your_solution} 'Dyalog APL' ⍝ show where the spaces are
0 0 0 0
0 0 1 0
0 0 1 1
1 1 1 1

      {your_solution} 100 ⍝ should return a 1×1 matrix
100

      {your_solution} ⍬  ⍝ should return a 0×0 matrix

      ⍴{your_solution} ⍬  ⍝ should return a 0×0 matrix
0 0
```