Dyalog APL Problem Solving Contest - Phase I Questions

1) ***Seems a Bit Odd to Me***
   Write a d-fn to produce a vector of the first n odd numbers.

   Test cases:
   {your_solution} 10 should produce 1 3 5 7 9 11 13 15 17 19
   {your_solution} 0 ⍬ this should return an empty vector
   {your_solution} 1 should return 1

2) ***Making the Grade***
   Write a d-fn which returns the percent (from 0 to 100) of passing (65 or higher) grades in a vector of grades.

   Test cases:
   {your_solution} 25 90 100 64 65 should return 60
   {your_solution} 50 should return 0
   {your_solution} 80 90 100 should return 100
   {your_solution} ⍳0 should return 100 (all grades in an empty vector are passing)

3) ***What's in a Word?***
   Write a d-fn which returns the number of words character vector. For simplicity's sake, you may consider the space character ' ' to be the only word separator.

   Test cases:
   {your_solution} 'Testing one, two, three' should return 4
   {your_solution} '' should return 0
   {your_solution} ' this vector has extra   blanks   ' should return 5 (extra blanks do not count).

4) ***Keeping Things in Balance***
   Write an APL d-fn which returns a 1 if the opening and closing parentheses in a character vector are balanced, or a zero otherwise.

Test cases:
{your_solution} '((2×3)+4)' should return 1
{your_solution} '' should return 1
{your_solution} 'hello world!' should return 1
{your_solution} ')(2×3)+4(' should return 0
{your_solution} '(()' should return 0
{your_solution} ')' should return 0

5) ***Identity Crisis***
An identity matrix is a square matrix (table) of 0 with 1's in the main diagonal. Write an APL d-fn which produces an n×n identity matrix.

Test cases:
{your_solution} 4 should produce:
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
{your_solution} 1 should return a 1×1 matrix:
1
{your_solution} 0 should return a 0×0 matrix

6) ***Home on the Range***
Write a d-fn which returns the magnitude of the range (i.e. the difference between the lowest and highest values) of a numeric array.

Test cases:
{your_solution} 19 ¯3 7.6 22 should return 25
{your_solution} 101 should return 0 (should work with a scalar argument)
{your_solution} 2 3⍴10 20 30 40 50 60 should return 50 (should work with arrays of any number of dimensions)
{your_solution} ⍳0 should return 0 (should work with empty arrays as arguments)

7) ***Float your Boat***
Write a d-fn which selects the floating point (non-integer) numbers from a numeric vector.

Test cases:
{your_solution} 14.2 9 ¯3 3.1 0 ¯1.1 should return 14.2 3.1 ¯1.1
{your_solution} 1 3 5 should return an empty vector
{your_solution} 3.1415 should return 3.1415

8) ***Go Forth and Multiply***
Write a d-fn which produces a multiplication table.

Test cases:
{your_solution} 4 should return
1 2 3 4

2 4 6 8
3 6 9 12
4 8 12 16
{your_solution} 1 should return a 1x1 matrix: 1
{your_solution} 0 should return a 0x0 matrix

9) It's a Moving Experience

Write a d-fn which produces n month moving averages for a year's worth of data.

Test cases:
sales←200 300 2700 3400 100 2000 400 2100 3500 3000 4700 4300
2 {your_solution} sales should produce 2 month moving averages: 250 1500 3050 1750 1050 1200 1250 2800 3250 3850 4500
10 {your_solution} sales should produce 10 month moving average: 1770 2220 2620
1 {your_solution} sales should produce 200 300 2700 3400 100 2000 400 2100 3500 3000 4700 4300 (1 month moving average is the same as sales)

10) Solution Salvation

Many people have taken some sort of algebra class where you are presented with a set of linear equations like:
3x + 2y = 13
x - y = 1
The answer in this case is x=3 and y=2

Write a d-fn which solves this type of problem. Hint: this is the easiest of all of the problems presented here. The left argument is a vector of the values for the equations and the right argument is a matrix of the coefficients.

Test cases:
13 1 {your_solution} 2 2⍴3 2 1 ¯1 should produce
3 2

2 6 4 {your_solution} 3 3⍴4 1 3 2 2 2 6 3 1 should produce
¯1 3 1