Dyalog APL - Assessment A

1. *Create a function* `MultiplyAndSkip` that accepts scalar numbers ⍺ and ⍵. Your function needs to return the first 7 integers multiplied by ⍺ and then added to ⍵:

```
      1 MultiplyAndSkip 0
1 2 3 4 5 6 7
      2 MultiplyAndSkip 0
2 4 6 8 10 12 14
      2 MultiplyAndSkip ¯3
¯1 1 3 5 7 9 11
      3 MultiplyAndSkip 1
4 7 10 13 16 19 22
```

2. Consider the function `Mat1`:

```
     Mat1←{2 3⍴⍳6}   ⍝ note: no "⍵" in Mat1

      Mat1 0          ⍝ The same result for any argument
1 2 3
4 5 6
      Mat1 2 3 4⍴42 6 ¯1
1 2 3
4 5 6
      Mat1 ⍬
1 2 3
4 5 6
```

   Create the functions Mat2 and Mat3:

```
      Mat2 0
¯1 0 1
 2 3 4
      Mat3 0
¯1 ¯2
¯3 ¯4
¯5 ¯6
```

3. *Create two functions*, F and G, that behave as follows:

```
      (2 3⍴1) F (2 3⍴2)
1 1 1
1 1 1
2 2 2
2 2 2
      (1 3⍴'ABC') F 2 3⍴'XYZ'
ABC
XYZ
XYZ
      G 3 3⍴⍳9
1 2 3 1 2 3
4 5 6 4 5 6
7 8 9 7 8 9
      G 3 1⍴'XYZ'
XX
YY
ZZ
```

4. *Create a function* `ToVector` that accepts any array as ⍵ and turns it into a vector with the same elements:

```
      ToVector 1 2 3 4
1 2 3 4
      ToVector 3 2⍴⍳6
1 2 3 4 5 6
      ToVector 2 2 2⍴⍳8
1 2 3 4 5 6 7 8
      ⍴ToVector 4 2
1
```

5. *Create a function* `Sum2ndLast` that accepts a numeric array as ⍵ and sums its contents along the 2nd to last axis:

```
      Sum2ndLast 2 3⍴⍳6
5 7 9
      Sum2ndLast 2 10 2⍴⍳40
100 110
300 310
      Sum2ndLast 2 2 10 2⍴⍳80
100 110
300 310

500 510
700 710
```

6. *Create a function* `DropRandRows` that accepts a matrix as `⍵` and drops, from the top, between 1 and all the rows of the matrix (chosen randomly).
   Here are some example runs:

```
      DropRandRows 3 4⍴⍳12
5  6  7  8
9 10 11 12
      DropRandRows 3 4⍴⍳12      ⍝ Empty result!
      DropRandRows 3 4⍴⍳12
9 10 11 12
      DropRandRows 3 4⍴⍳12
9 10 11 12
      DropRandRows 5 3⍴⎕A
JKL
MNO
      DropRandRows 5 3⍴⎕A
MNO
      DropRandRows 5 3⍴⎕A
GHI
JKL
MNO
```

7. *Create a function* `PickRandCols` that accepts a matrix as ⍵ and takes, from the left, between 1 and all columns from the left (chosen randomly).
   Here are some example runs:

```
      PickRandCols 2 5⍴⍳10
1 2 3
6 7 8
      PickRandCols 2 5⍴⍳10
1
6
      PickRandCols 2 5⍴⍳10
1 2 3 4
6 7 8 9
      PickRandCols 3 10⍴⎕A
ABCD
KLMN
UVWX
      PickRandCols 3 10⍴⎕A
ABCDEFGHIJ
KLMNOPQRST
UVWXYZABCD
      PickRandCols 3 10⍴⎕A
ABCDEF
KLMNOP
UVWXYZ
```

8. *Create a function* `RIota` that takes a positive integer `⍵` and produces the same integers as `⍳⍵`, but in the reverse order:

```
      RIota 5
5 4 3 2 1
      RIota 1
1
      RIota 10
10 9 8 7 6 5 4 3 2 1
```

9. *Create a function* `Reverse` *which uses* `RIota` in its definition. It should take a vector `⍵` and returns the same vector in the reverse order:

```
      Reverse ⍳5
5 4 3 2 1
      Reverse 'Hello, world!'
!dlrow ,olleH
      Reverse 15 30 2 8
8 2 30 15
```

10. *Create a function* `ColumnsSurpass` that accepts a scalar number as `⍺` and a numeric matrix as `⍵`, and returns all columns of `⍵` whose sum is strictly greater than `⍺`:

```
      40 ColumnsSurpass 4 5⍴⍳20
 3  4  5
 8  9 10
13 14 15
18 19 20
      48 ColumnsSurpass 4 5⍴⍳20
 5
10
15
20
       8 ColumnsSurpass 2 8⍴3 5 12 5 3 7 1
 5 12 3
12  5 7
       16 ColumnsSurpass 2 8⍴3 5 12 5 3 7 1
 5 12
12  5
       ⍴17 ColumnsSurpass 2 8⍴3 5 12 5 3 7 1
2 0
```

11. *Define a function* `Range` that accepts a numeric vector `⍵` and returns the range of `⍵`. That is, it returns the difference between the largest element of `⍵` and the smallest element of `⍵`:

```
      Range ⍳10
9
      Range 0 100
100
      Range 100 0
100
      Range 4 ¯3 4 5
8
      Range 15 64 23 ¯64 ¯15
128
```

12. *Create a function* `MaxRangeRows` that accepts a numeric matrix `⍵` and returns the row with the largest range:

```
       MaxRangeRows 2 3⍴0 10 5 1 2 3
0 10 5
       MaxRangeRows 2 3⍴1 2 3 10 0 5
10 0 5
       MaxRangeRows 5|2 3⍴⍳6
4 0 1
       MaxRangeRows 5 2⍴3 5 ¯4 2 ¯3
5 ¯4
       MaxRangeRows 4 3⍴0 ¯3 5 7
 7 0 ¯3
¯3 5  7
```

13. *Create a function* `IsInteger` that takes a numeric vector `⍵` and returns 1 if `⍵` is an integer, and `0` otherwise:

```
      IsInteger 3
1
      IsInteger 3.3
0
      IsInteger ¯3 3 3.3
1 1 0
      IsInteger (⍳10)÷2
0 1 0 1 0 1 0 1 0 1
```

14. *Write a function* `TopDown` that accepts a numeric matrix `ω` and reorders its rows so that the row with the largest number shows up first and the row with the smallest largest number shows at the bottom:

```
      TopDown 4 3⍴0 12 9 8 1 2 3 4 10 5 6 7
0 12  9
3  4 10
8  1  2
5  6  7
      TopDown 3 3⍴⍳9
7 8 9
4 5 6
1 2 3
       TopDown 3 2⍴1 0 ¯1 4 3
¯1 4
 3 1
 1 0
```

15. *Write a function* `RemoveFrom` that accepts a vector `⍺` and removes those items from the vector `ω`:

```
      'abc' RemoveFrom 'cabana'
n
      'abc' RemoveFrom 'cape'
pe
      1 2 3 RemoveFrom 0 1 0 2 0 45
0 0 0 45
```

16. *Write a function* `RemoveExtra` that accepts a vector `ω` and returns the same vector after removing all occurrences of its most common elements:

```
      RemoveExtra 'banana'
bnn
      RemoveExtra 2|⍳7
0 0 0
      RemoveExtra 'Mississippi'
Mpp
     RemoveExtra 'aaeeoox'
X
```

17. *Write the function* `TimesTable` to generate the multiplication table for numbers up to ⍵:

```
      TimesTable 3
1 2 3
2 4 6
3 6 9
      TimesTable 5
1  2  3  4  5
2  4  6  8 10
3  6  9 12 15
4  8 12 16 20
5 10 15 20 25
```

18. *Write the function* `IsSorted` which determines if a given array is already in ascending order:

```
      IsSorted 3 1 4 1 5
0
      IsSorted 2 3 5 7 11
1
      IsSorted 3 4⍴'Bob Abe Carl'
0
      IsSorted 3 4⍴'Abe Bob Carl'
1
```

19. *Write the function* `AnyCopies` which determines if the vector ⍵ has any duplicates:

```
      AnyCopies 'India'
0
      AnyCopies 'Indian'   ⍝ there are 2 "n"s
1
```

20. *Write the function* `Trim` which removes leading spaces from the vector ⍵:

```
      Trim '     poof'
poof
      Trim '   I have spaces'
I have spaces
      Trim 'nospace'
Nospace
```

21. *Write the function* `KeepOnly` which removes from the vector ⍺ any element not in ⍵:

```
      'Hello World' KeepOnly ⎕A
HW
      3 1 4 1 5 KeepOnly ⍳3
3 1 1
      'here_we_go' KeepOnly '_w'
_w_
```

22. *Write the function* `HasElements` which determines if the array `⍵` has any elements (i.e. it isn't empty):

```
      HasElements 3 1 4
1
      HasElements ⍳0
0
      HasElements 3 2 4⍴⎕A
1
      HasElements 3 0 4⍴⎕A
0
      HasElements 42
1
```

23. *Write the function* `Overlaps` which determines if the arrays `⍺` and `⍵` have any elements in common.

```
      2 7 1 8 Overlaps 3 1 4
1
      1 2 3 Overlaps 7 8 9 10
0
```

24. *Write the function* `OfLength` which takes a matrix `⍺` and returns the rows that have exactly `⍵` letters. Remove any spaces from the result.

```
      names←7 6⍴'Patel Arya  Babu  Dewan Singh GandhiGupta '
      names OfLength 5
Patel
Dewan
Singh
Gupta
      ⍴names OfLength 5
4 5
      names OfLength 6
Gandhi
      ⍴names OfLength 6
1 6
```

25. *Write the function* `Explode` which generates a vector consisting of one 1, two 2s, three 3s, and so on, until the argument number:

```
      Explode 5
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
      Explode 3
1 2 2 3 3 3
      Explode 1
1
      Explode 0
0
```

26. *Write the function* `NoFizzBuzz` which removes any elements from the vector `⍵` which are divisible by 3 or 5.

```
      NoFizzBuzz ⍳10
1 2 4 7 8
      NoFizzBuzz 2 4 6 8 10 12
2 4 8
      NoFizzBuzz 2 4 6 8 ¯10 12   ⍝ don't forget negatives!
2 4 8
```

27. *Write the function* `CentreIn` which takes a character vector `⍺` and adds spaces on the left and right so it becomes `⍵` characters long, with the original text approximately centred. The number of added spaces on the left and right must not differ by more than 1. In the examples below, we have marked space characters in shades of grey for clarity.

```
      'Boo' CentredIn 5
 Boo
      ⍴'Boo' CentredIn 5
5
      'Boom' CentredIn 5   ⍝ returning 'Boom ' is also OK
 Boom
      'Hi' CentredIn 10
   Hi
```

28. *Save your workspace*, with a name like:
    assessmentA_Your_Name.dws
    Email your workspace file to workshops@dyalog.com with a subject like:
    assessment A your name