Task 1

Download the workspace `assessmentB.dws` and load it.

This workspace contains functions and variables which will be used in some of the tasks for this assessment.

Save the workspace as `assessmentB_your_name.dws`

Task 2

Create a function `EveryN` which:

- Takes a scalar left argument `n`
- Takes a vector right argument `v`
- Returns a vector containing only every `n`th element of `v`

```
      3 EveryN ⎕A
ADGJMPSVY
      5 EveryN ⎕A
AFKPUZ
      2 EveryN ⍳20
1 3 5 7 9 11 13 15 17 19
```

Task 3

Rewrite these three functions so that they do not contain parentheses ( )

```
Task3A
Task3B
Task3C
```

Do not change the names of the functions.

Task 4

The variable `S6124735` contains the result of an indexing expression:

```
        S6124735 ← S[6 1 2 4 7 3 5]
```

Define the variable S in your workspace.

Task 5

In the workspace is a 1-element vector `v1`.

Modify the variable `v1` so that it is a scalar with the same value.

Task 6

Create the function `Compound10k` which

- Takes a left argument integer vector of percentage interest rates
- Takes a right argument integer vector of years
- Returns a table showing the value of a $`10000` investment that grows (each year) by the given percentages, at the years specified

```
      2 5 8 Compound10k ⍳7
10200 10404 10612.08 10824.3216 11040.80803 11261.62419 11486.85668
10500 11025 11576.25 12155.0625 12762.81563 13400.95641 14071.00423
10800 11664 12597.12 13604.8896 14693.28077 15868.74323 17138.24269

      3 9 Compound10k 3 6 9 12
10927.27 11940.52297 13047.73184 14257.60887
12950.29 16771.00111 21718.93279 28126.64782

      ⌊0.5+ 2.3 12 Compound10k 0,5×⍳4
10000 11204 12553 14065 15758
10000 17623 31058 54736 96463
```

Task 7

Create a variable `nmat` which is a simple numeric array for which this works:

```
      +/nmat
7 17
      +⌿nmat
8 13 3
      nmat[1;2]
4
      nmat[2;3]
6
```

Task 8

Create a variable `nested` which is a nested array for which this works:

```
      ⍴nested
3
      ⊃nested
5
      ⊃⌽nested
21 22
23 24
      ]disp +/nested
┌─────┐
│28 35│
│34 37↓
└~───→┘
```

Task 9

In the workspace is a variable `words` which contains a dictionary of 99119 words.

Create a function `WordSummary` which takes a nested vector of character vector as its right argument and returns a nested vector. Each element in the nested vector result is computed as follows:

1. The number of palindromes
2. The number of words containing at least 3 `'e'`s. For example: `'aberdeen'` and `'eileen'`
3. The number of words containing 2 consecutive `'e'`s. For example: `'aberdeen'`, `'eileen'` and `'reel'`
4. The number of words in which the letters are already in alphabetical order. For example: `'empty'` and `'all'`
5. The length of the longest palindrome(s)

```
      ]disp WordSummary 'aberdeen' 'empty' 'eve' 'eileen' 'reel' 'racecar' 'all'
┌→┬─┬─┬─┬─┐
│2│2│3│2│7│
└─┴─┴─┴─┴→┘
      ]disp WordSummary words
┌→─┬────┬────┬───┬─┐
│57│3959│2149│388│7│
└~─┴~───┴~───┴~──┴→┘
```

Task 10

Create a function `HeightConverter` which:

- Takes a character vector left argument
- Takes a numeric array right argument
- Converts distances between metres and inches and feet. (1 inch is 0.0254 metres and 1 feet is 12 inches.)

```
      'm→in' HeightConverter 1.7
5 6.929133858
      'm→in' HeightConverter 1.85
6 0.834645669З
      'in→m' HeightConverter 6 1
1.8542
      'in→m' HeightConverter 5 11
1.8034
```

Task 11

Create a function `RotationOf` which:

- Takes a left vector argument
- Takes a right argument that is a nested vector of vectors
- Returns the elements of the right argument that are rotations of the left argument

```
      ]disp 'racecar' RotationOf 'racecar' 'carrace' 'rraacce'
┌→──────┬───────┐
│racecar│carrace│
└──────→┴──────→┘
      ]disp 'oi' RotationOf 'oi' 'io' 'oi' 'io' 'ii' 'oo'
┌→─┬──┬──┬──┐
│oi│io│oi│io│
└─→┴─→┴─→┴─→┘
       ]disp 'pepper' RotationOf 'water' 'salt' 'oil'
┌⊖─────┐
│      │
└─────→┘
       ]disp 1 2 3 RotationOf (1 2 3)(2 3 1)(3 1 2)(3 2 1)
┌→────┬─────┬─────┐
│1 2 3│2 3 1│3 1 2│
└~───→┴~───→┴~───→┘
```

Task 12

The Collatz sequence is a mathematical sequence that works as follows:

- you start with a positive integer
- if the integer is even, you divide it by 2
- if the integer is odd, you multiply by 3 and add 1
- you repeat the steps above until you reach the number 1

For example, starting with `5`, we do:

```
5 → 16 (1+3×5) → 8 (16÷2) → 4 (8÷2) → 2 (4÷2) → 1 (2÷2)
```

Create the integer vector `collatzPath21` that contains the path we do if we start with `21`. When you are done, the following should work:

```
      ⍴collatzPath21
8
      ⊃collatzPath21
21
      ⊃⌽collatzPath21
1
```

Task 13

The workspace contains a function `GOTOCollatz` that is currently buggy. Fix the issues, so that it behaves as desired:

```
      GOTOCollatz 1
1
      GOTOCollatz 5
5 16 8 4 2 1
      GOTOCollatz 35
35 106 53 160 80 40 20 10 5 16 8 4 2 1
      GOTOCollatz 38
38 19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

NOTE: if a function seems to be running forever, use the menu "Action → Interrupt" to stop the function.

Task 14

Create a function `Collatz` which:

- Takes a positive integer right argument
- Returns the integers in the Collatz path of the argument (that is, behaves like `GOTOCollatz` from the previous task)
- Uses recursion to build the path

```
      Collatz 1
1
      Collatz 5
5 16 8 4 2 1
      Collatz 21
21 64 32 16 8 4 2 1
      Collatz 38
38 19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

NOTE: a recursive function is a function that calls itself. Here are two recursive examples that do the same thing as the `!` primitive:

```
Fact ← {ω=0: 1 ⋄ ω×∇ω-1} ⍝ recursive dfn

∇ fact ← Fact n       ⍝ recursive tradfn
    :If n=0
        fact ← 1
    :Else
        fact ← n×Fact n-1
    :EndIf
∇
```

Submit Your Workspace

Save your workspace, with a name like:
assessmentB_Your_Name.dws

Email to workshops@dyalog.com with a subject like:
assessment B your name

You can submit any time up to Thursday 9th September 13:00 IST