

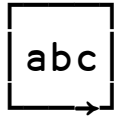
Task 1

1.) CLEAR to clear your workspace
2.) SAVE your workspace with a workspace ID like **tasks8_your_name.dws**

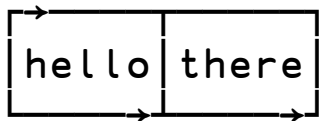
Task 2

Write a function `EncloseIfSimple` which returns its argument array with an extra level of nesting, if its current nesting is less than or equal to one.

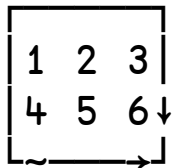
```
]disp EncloseIfSimple 'a' 'b' 'c'
```



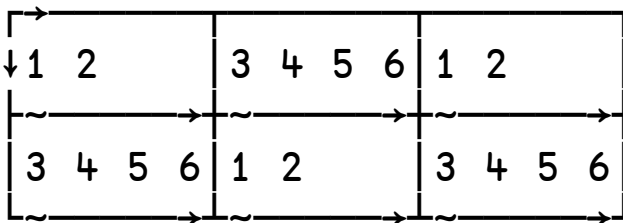
```
]disp EncloseIfSimple 'hello' 'there'
```



```
]disp EncloseIfSimple 2 3p(1 2),2+14
```



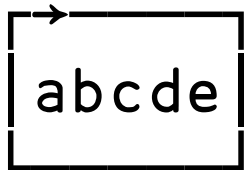
```
]disp EncloseIfSimple 2 3p(1 2)(2+14)
```



Task 3

Create a variable `charvec` which is a simple character vector containing the first five letters of the alphabet in alphabetical order.

```
]display charvec
```

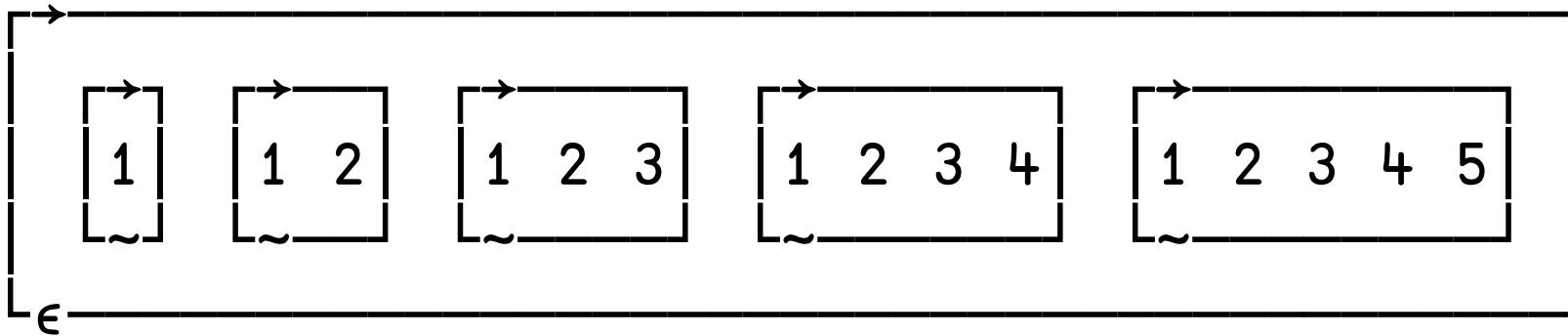
A diagram representing a character vector. It consists of a rectangular box with a black border. Inside the box, the letters 'a', 'b', 'c', 'd', and 'e' are written in a monospaced font. Above the top-left corner of the box, there is a small black arrow pointing to the right, indicating the start of the vector.

abcde

Task 4

Create a variable `iotas` which is a 5-element nested vector of numeric vectors, each of which is a list of integers from 1 to that vector's index in `iotas`.

```
]display iotas
```



Task 5

Create a variable `little_nest` which has the following properties:

`plittle_nest`

2 3

`≡ little_nest`

-2

`p"little_nest`

		2
3		6

`]display elittle_nest`

→
[I 3 am 1 5 8 amatrix]
+

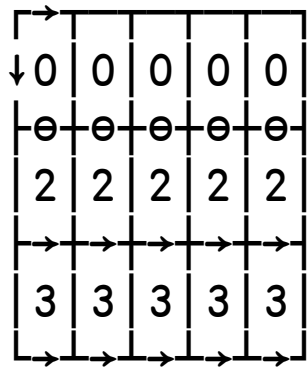
`pelittle_nest`

14

Task 6

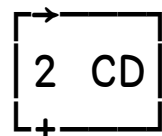
Create a variable `big_nest` which is a nested 3 by 5 matrix. The first row contains numeric scalars 1 2 3 4 5. The second row contains five two-element character vectors 'AB' 'CD' 'EF' 'GH' 'XY' and the third row contains the concatenation of the numbers in the first row and character vectors from the second row.

```
]disp ρ`big_nest
```



0	0	0	0	0
0	0	0	0	0
2	2	2	2	2
3	3	3	3	3

```
]display (c3 2)>big_nest
```



2	CD
+	

Task 7

Write a function `AllEach` that takes a nested array of Boolean vectors as its argument.

It returns a simple (non-nested) Boolean array of the same shape as its argument, where 1 indicates all 1s in the argument vector and 0 otherwise.

```
]display AllEach 1(1 0)(1 1 1)(1 1)(1 1 1 0)
```

```
→  
[1 0 1 1 0]
```

```
]display AllEach 2 3p(1 0 0)(1 1)(1 0 1 0 0)(1 1 1 1 1)
```

```
→  
↓0 1 0  
[1 0 1]
```

```
]display AllEach 3 1 5p(1 0 0)(1 1)(1 0 1 0 0)(1 1 1 1 1)
```

```
→  
↓↓0 1 0 1 0  
[1 0 1 0 1  
0 1 0 1 0]
```

Task 8

Create a function `ReplaceHead` which returns its left argument vector α , but with the first $\rho\omega$ elements replaced with the contents of ω :

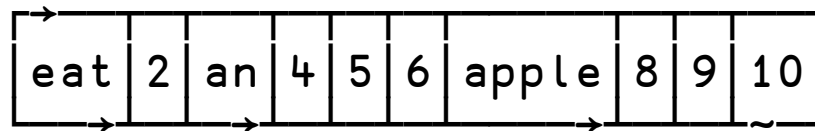
```
'apple' ReplaceHead 'Eat '  
Eat le  
'apple' ReplaceHead 'rang'  
range  
'apple' ReplaceHead 'ENTERPRISE '  
ENTER
```


Task 9

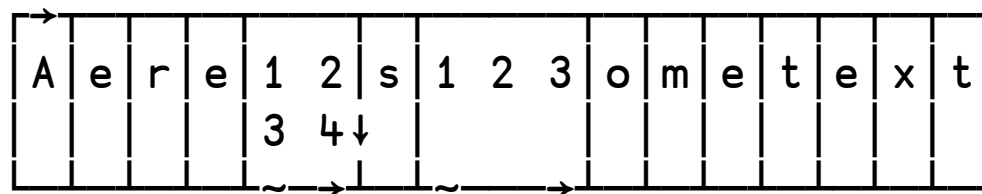
Create a function `ReplaceAt` which returns its right argument vector, except with elements at indices specified by the 1st element of its left argument replaced by items given in the 2nd element of its left argument:

```
]disp (1 5 7) 'ABC' ReplaceAt 'hereissometext'
AereBsCometext
```

```
]disp (1 3 7)('eat' 'an' 'apple') ReplaceAt 110
```



```
]disp (1 5 7) ('A' (2 2 1 4) (1 2 3)) ReplaceAt 'hereissometext'
```



Task 10

Create a function `ReplaceRow` which returns an array the same shape as its left argument, except that the row specified in the 1st element of its right argument is replaced with the 2nd element. If the 2nd element has the correct length, it is distributed throughout the row.

```
]disp (2 3p12) ReplaceRow 2 ('yo')
```

↓1	2	3
~	~	~
yo	yo	yo

```
]disp (4 3p12) ReplaceRow 2 ('you')
```

```
1  2  3
y  o  u
7  8  9
10 11 12
```

```
]disp (2 3p12) ReplaceRow 2 (14)
```

↓1	2	3
~	~	~
1 2 3 4	1 2 3 4	1 2 3 4

```
]disp (2 3p12) ReplaceRow 2 ((1 1)(5 5)(9 9))
```

↓1	2	3
~	~	~
1 1	5 5	9 9

Task 11

Create a function `MaskAt` which returns a Boolean vector with 1s at indices specified by its right argument vector:

```
MaskAt 1 3 5  
1 0 1 0 1
```

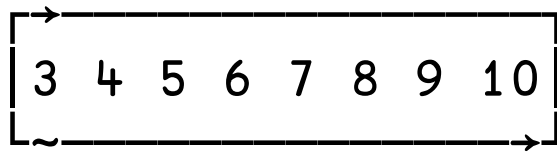
```
MaskAt 1 7 2  
1 1 0 0 0 0 1
```

```
MaskAt 7 7 3 8  
0 0 1 0 0 0 1 1
```

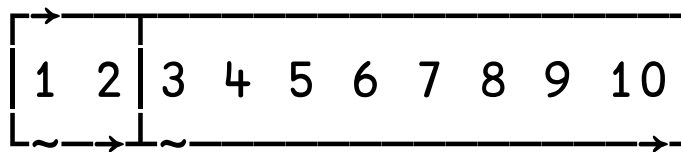
Task 12

Create a function `SplitAt` which partitions its simple argument vector into a nested vector of vectors, with partitions beginning just before indices specified by its left argument vector:

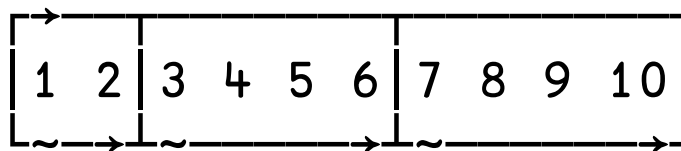
```
]disp 3 SplitAt v10
```



```
]disp 1 3 SplitAt v10
```



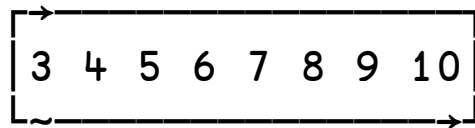
```
]disp 1 3 7 SplitAt v10
```



Task 13

Create a function `SplitWhere` which partitions its simple argument vector into a nested vector of vectors, with new partitions beginning just before occurrences of elements given in its left argument vector:

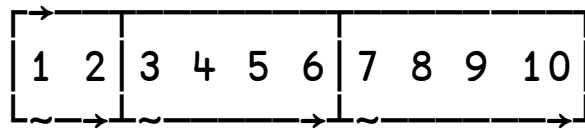
```
]disp 3 SplitWhere 1 10
```



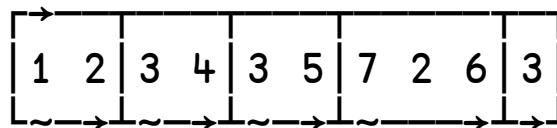
```
]disp 1 3 SplitWhere 1 10
```



```
]disp 1 3 3 7 SplitWhere 1 10
```



```
]disp 1 3 7 SplitWhere 1 2 3 4 3 5 7 2 6 3
```



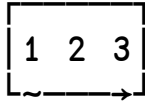
Task 14

Create a function `EncloseRows` which encloses the last axis of its argument array, or the whole array if it is a vector or scalar:

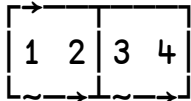
```
]disp EncloseRows 42
```

42

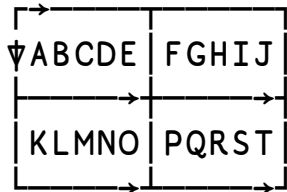
```
]disp EncloseRows 1 2 3
```



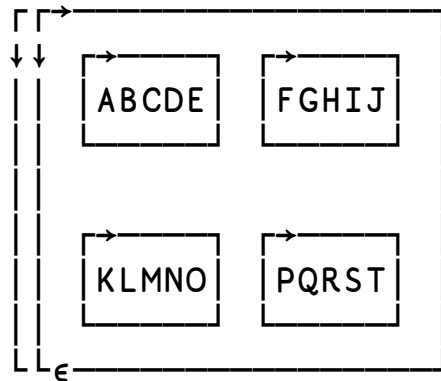
```
]disp EncloseRows 2 2pi4
```



```
]disp EncloseRows 2 1 2 5pA
```



```
]display EncloseRows 2 1 2 5pA
```



NOTE: Enclosing a simple scalar returns that scalar. **BONUS:** Find the primitive function (symbol) which is the same function.

Task 15

You may use the `Part` function in your solution:

$$\text{Part} \leftarrow \{\Box \text{ML} \leftarrow 3 \ \diamond \ \alpha \subset \omega\}$$

Create a function `SplitByDelimiters` which partitions its right argument vector at locations where elements in its left argument are found.

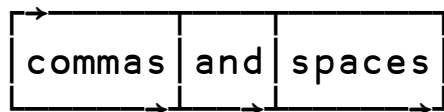
```
]disp ' ' SplitByDelimiters 'space separated text'
```



```
]disp ', ' SplitByDelimiters 'comma,separated,2,3,values'
```



```
]disp ', ' SplitByDelimiters 'commas and,spaces'
```



BONUS: Can you create similar behaviour in a function `SplitEnclose` which uses the partitioned-enclose function (\subset when $\Box \text{ML} \leftarrow 1$) instead of the `Part` function.

Task 16

Create a function NumList which converts a simple character scalar or vector of digits into a numeric vector:

```
]display NumLists '42'
```

A MATLAB array representation of a 1x2 numeric vector. It consists of a square bracket with a right-pointing arrow above it and a tilde (~) below it. Inside the bracket, the numbers 4 and 2 are displayed.

```
]display NumLists '3'
```

A MATLAB array representation of a 1x1 numeric vector. It consists of a square bracket with a right-pointing arrow above it and a tilde (~) below it. Inside the bracket, the number 3 is displayed.

```
+ / NumList '12,14 -3 55'
```

78

```
]display NumList '1 3 4 5'
```

A MATLAB array representation of a 1x4 numeric vector. It consists of a square bracket with a right-pointing arrow above it and a tilde (~) below it. Inside the bracket, the numbers 1, 3, 4, and 5 are displayed.

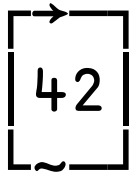
```
]display NumList '12,23 5 4 3,6,2,15'
```

A MATLAB array representation of a 1x8 numeric vector. It consists of a square bracket with a right-pointing arrow above it and a tilde (~) below it. Inside the bracket, the numbers 12, 23, 5, 4, 3, 6, 2, and 15 are displayed.

Task 17

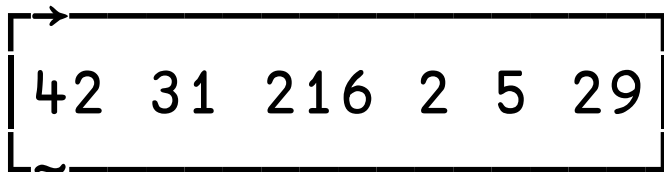
Create a function HashNums which converts a character scalar digit, or vector of digits separated by octothorpes ('#' characters, sometime called "hash") into a numeric vector.

```
]display HashNums '42'
```



```
[42]
```

```
]display HashNums '42#31#216#2#5#29'
```

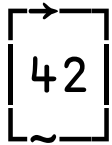


```
[42 31 216 2 5 29]
```

Task 18

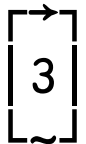
Create a function `NumLists` which converts a character scalar digit, or vector of digits, spaces and commas, and returns a numeric vector or nested vector of numeric vectors. Spaces in the argument separate individual scalar numbers, while commas separated lists of numbers:

```
]display NumLists '42'
```



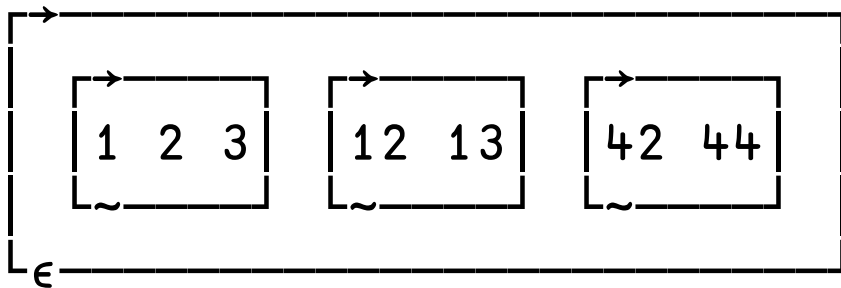
A diagram representing a numeric vector. It consists of a vertical rectangle with a right-pointing arrow at the top and a tilde (~) at the bottom. Inside the rectangle is the number 42.

```
]display NumLists '3'
```



A diagram representing a numeric vector. It consists of a vertical rectangle with a right-pointing arrow at the top and a tilde (~) at the bottom. Inside the rectangle is the number 3.

```
]display NumLists '1 2 3,12 13,42 44'
```



A diagram representing a nested vector structure. It consists of a large vertical rectangle with a right-pointing arrow at the top and a tilde (~) at the bottom. Inside this rectangle, at the bottom left, is a small 'e' symbol. To the right of the 'e' are three smaller vertical rectangles, each with a right-pointing arrow at the top and a tilde (~) at the bottom. The first small rectangle contains the numbers 1 2 3. The second small rectangle contains the numbers 12 13. The third small rectangle contains the numbers 42 44.

Task 19

Write a function `RowOPTimes` which multiplies all combinations of rows from its left argument and right argument numeric arrays.

```

      ρ(3 5ρ5/10×ι3) RowOPTimes 2 5ρ(10ρ1 0)\(⁻1+2×ι5)
3 2 5
      (3 5ρ5/10×ι3) RowOPTimes 2 5ρ(10ρ1 0)\(⁻1+2×ι5)
10  0 30  0 50
 0 70  0 90  0

20  0 60  0 100
 0 140  0 180  0

30  0 90  0 150
 0 210  0 270  0

```

```

      ρ(2 2ρ1 3 2 5 3) RowOP 2 3
2 2
      (2 2ρ1 3 2 5 3) RowOP 2 3
2 9
4 15
      ρ(2 1 5ρ1 3 2 5 3) RowOP 2 5ρ2 3 5 2
2 1 2 5
      (2 1 5ρ1 3 2 5 3) RowOP 2 5ρ2 3 5 2
2 9 10 10 6
3 15 4 10 9

2 9 10 10 6
3 15 4 10 9

```

Submit Your Workspace

Save your workspace, with a name like:

tasks8_your_name.dws

Email to workshops@dyalog.com with a subject like:

Tasks 8 Your Name