Define the Worksheet Index Origin - Must Be Done Before Defining These Functions!!!!

$$\mathbf{ORIGIN} := 9$$

- **vec(z)**: *vec* takes a single argument and converts it, appropriately, into a vector. In particular, it converts a scalar into a single-element array, and stacks the columns of a matrix into a vector. It works with nested arrays and will also convert range definitions into a vector with the corresponding values. In M13..M15, vec will convert a range variable into a vector, provided that vec is on the right-hand side of a definition.
- This usage is in keeping with **use of vec in matrix theory** and extended to cover scalars and nested arrays.

```
vec(z) ≡ v_ORIGIN ← 0
         for e ∈ z
           if IsArray(e)
             v ← stack(v, vec(e))
           else
             v_(last(v)+1) ← e
         submatrix(v, 1 + ORIGIN, last(v), ORIGIN, ORIGIN)
```

- **seq(z):** *seq* takes a single argument ***z*** and converts it, appropriately, into a vector containing a sequence of values. In particular, it converts a scalar into a sequence of the form 0..***z***-1, unless ***z***=0 in which case it returns 0 (representing the empty array) otherwise it behaves like vec.

```
seq(x) ≡ if IsScalar(x)
           if x
             for k ∈ ORIGIN..x
               v_k ← k − ORIGIN
             v
           else
             0
         else
           vec(x)
```

- **index(v):** creates a vector or a nested vector giving the index of each element of ***v***; useful for plotting ***v***. If ***v*** is a scalar, then **index** returns a list of sequential integers up to and including ***v***-1. If v is a 2D array, then index returns a nested 2-vector containing the row and column indices respectively.

```
index(x) := if IsScalar(x)
              if x
                (seq(x − 1) + ORIGIN)
              else
                0
            else
              if IsArray(x)
                if rows(x) = 1
                  index(cols(x))
                else
                  if cols(x) = 1
                    index(rows(x))
                  else
                    [ index(rows(x)) ]
                    [ index(cols(x)) ]
              else
                x
```

$$index(3) = \begin{bmatrix} 9 \\ 10 \\ 11 \end{bmatrix}$$

Ooh! Cat Skinning Time! We know a song about that, don't we children?
It's nice to be able to pick a set of values from a vector if we know the indices that we want. So let's define a function to do that.

- **pick(*v*,*lst*):** takes a vector ***v*** and a list of integer indices ***lst*** and returns a vector of the indexed values of ***v***. If ***v*** is a scalar **pick** converts it to a single-element vector (but **pick** will only return [***v***] if ***lst*** = ORIGIN. Returns 0 if lst does not contain any valid indices.

```
pick(v, lst) :=
    if IsScalar(v)
        stack(v)
    w ← 0
    k ← ORIGIN − 1
    for i ∈ lst
        try
            w_{k ← k+1} ← v_i
        on error
            continue
    w
```

$pick(seq(5),2)=0$

$seq(5)^{T}=[0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5]$

Now we can rewrite Mike's mid index function and combine it with Werner's ORIGIN independence

```
midindex(v) :=
    k ← floor((ORIGIN + last(v)) / 2)
    if mod(rows(v), 2) = 0
        k ← stack(k, k + 1)
    k
```

used floor instead of trunc as trunc rounds to zero, which makes negative ORIGIN behave differently to positive ORIGIN

Which let's us compose a new function that selects the middle indices and passes them straight to pick

$mid(v):=pick(v,midindex(v))$

$\mathbf{ORIGIN}=9$

And now get your spoons ready, children. It's --- Proof Of Pudding Time!

$v1:=seq(6)+11$

$\text{augment}(index(v1),v1)^{T}=\begin{bmatrix}9&10&11&12&13&14&15\\11&12&13&14&15&16&17\end{bmatrix}$

$\text{last}(v1)=15$

$midindex(v1)=12$

$mid(v1)=[14]$

$v2:=seq(7)+11$

$\text{augment}(index(v2),v2)^{T}=\begin{bmatrix}9&10&11&12&13&14&15&16\\11&12&13&14&15&16&17&18\end{bmatrix}$

$\text{last}(v2)=16$

$midindex(v2)=\begin{bmatrix}12\\13\end{bmatrix}$

$mid(v2)=\begin{bmatrix}14\\15\end{bmatrix}$

$index(v1)=\begin{bmatrix}9\\10\\11\\12\\13\\14\\15\end{bmatrix}$