The Vector Library

Mark Tarver, August 10, 2013

Functions in red are *compacting* - all unoccupied indices in the argument vectors are removed in the result.

vector. apply
((list A) --> B) --> (vector A) --> B
Apply a 1-place list processing function to a vector with analogous results.

vector. build
((list A) --> (list A)) --> (vector A) --> (vector A)
Apply a 1-place list processing function to a vector with analogous results.

vector. combine
((list A) --> (list A) --> (list A)) --> (vector A) --> (vector A) --> (vector A)
Apply a 2-place list processing function to 2 vectors with analogous results

```
(19+) (vector.apply (element? 3) (@v 1 2 3 <>))
true : boolean

(20+) (vector.build (function reverse) (@v 1 2 3))
<3 2 1> : (vector number)

(21+) (vector.combine (function append) (@v 1 2 3 <>) (@v 4 5 6 <>))
<1 2 3 4 5 6> : (vector number)
```

vector. blankv
(vector A) --> (vector B)
Create a blank vector of the same size as the input.

```
(21+) (vector.blankv (@v 1 2 3 <>))
<... ... ...> : (vector A)
```

vector.copy
(vector A) --> (vector A)
Copies a vector

vector.extend
(vector A) --> number --> (vector A)
Copies a vector, extending it by *n* indices.

```
(22+) (vector.copy (@v 1 2 3 <>))
<1 2 3> : (vector number)

(23+) (vector.extend (@v 1 2 3 <>) 5)
<1 2 3 ... ... ... ... ...> : (vector number)
```

vector.dmapv
(A --> A) --> (vector A) --> (vector A)
Destructively map a function along a vector.

vector.mapv
(A --> A) --> (vector A) --> (vector A)
Non-destructively map a function along a vector.

```
(25+) (dmapv (+ 1) (@v 1 2 3 <>))
<2 3 4> : (vector number)
```

vector.vector->list
(vector A) --> (list A)
Converts a vector to a list.

vector.list->vector
(list A) --> (vector A)
Converts a list to a vector

```
(23+) (vector.vector->list (@v 1 2 3 <>))
[1 2 3] : (list number)

(24+) (vector.list->vector [1 2 3])
<1 2 3>
```