Data Aggregates

A. The need for data aggregates

It is extraordinarily helpful to be able to refer to aggregates of data. Suppose you want to extract the upstream region of cpcB (the first gene of the operon devoted to formation of phycocyanin) from all cyanobacteria. You've found the names of all available cpcB genes. Now, you could get the sequences this way, one tedious gene at a time:

```
(DEFINE upstream-of-A7120-cpcB AS (SEQUENCE-UPSTREAM-OF alr0528))
:: "GGAACAGAGAGCAAGGGGGAGAAACTTAAAGATTTCCCCAATCCCAATTCCTCATCTCTCATCC..."

(DEFINE upstream-of-S6803-cpcB AS (SEQUENCE-UPSTREAM-OF sll1577))
:: "GACTAAAAAAAGACTTGAATGTCACTAACTACATCCAGTCTTTGCCATGGCCAGTCTTTCCACC..."

(DEFINE upstream-of-Npun-cpcB AS (SEQUENCE-UPSTREAM-OF npf5289))
:: "TAACCACAAGTTTGCAAAATTACAGATATACCTGTATAAATCAAAAAACTACGGCAATTTTATC..."
...
```

... and on and on. What is merely tedious here is a practical impossibility if you want to apply an operation to, say, every gene in an organism!

You *can* do operations one item at a time, but it is much more satisfying do them all at once. Here's how the operations could be combined:

```
(DEFINE cpcB-orthologs AS {alr0528 sll1577 npf5289 Av?5998 Te?5392} )
:: (ALR0528 SLL1577 NPF5289 AV?5998 TE?5392)

(SEQUENCES-UPSTREAM-OF cpcB-orthologs))
:: ("GGAACAGAGAGCAAGGGGGAGAAACTTAAAGATTTCCCCAATCCCAATTCCTCATCTCTCATCC..."
    "GACTAAAAAAAGACTTGAATGTCACTAACTACATCCAGTCTTTGCCATGGCCAGTCTTTCCACC..."
    "TAACCACAAGTTTGCAAAATTACAGATATACCTGTATAAATCAAAAAACTACGGCAATTTTATC..."
    ...)
```

B. Aggregate types: Strings, Lists, and Tables

Strings are collections of characters in a given order. They are delimited by double quotation marks. Here are examples of strings:

```
"Hello there"

(SEQUENCE-OF ssr1600) --> "TTGACGGTCAGTTTGCGC..."
```

Lists are collections of things (any type). They are delimited by curly brackets. Here are some examples of lists:

```
{Npun Avar A7120}

{"A" 1 "B" 2 "C" 3}

(ITEMS {.From .To .Direction} IN-EACH {all4312 npf0001 ssr1600} )
   --> ((5166997 5167767 :B) (167 1546 :F) (2024126 2024476 :F))
```

When lists are displayed by BioBIKE as the result of a function, they are delimited by parentheses rather than curly brackets. While you can't have strings within strings, you ***can*** have lists within lists, as shown in the result of the last example.

Tables are collections of things (any type) organized by subscripts or ***labels***. It is helpful to visualize them as tables with columns and rows (although they may be of any dimensionality). Here's an example of a two-dimensional table:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
nirA	A	A	T	T	T	T	G	T	A	G	C	T	A
ntcB	A	A	A	G	C	T	G	T	A	A	C	A	A
glnA	A	A	A	T	C	T	G	T	A	A	C	A	T
devB	C	G	T	T	C	T	G	T	A	A	C	A	A
urt	A	A	T	T	T	A	G	T	A	T	C	A	A

The labels are shown in bold, with row-labels being "nirA", "ntcB", etc, and the column-labels 1 through 13.

While the information within each cell usually is of the same type, it doesn't have to be. You could build a table looking like this:

	Size	**Gene-number**	**Habitat**
Npun	9059191	7412	"terrestrial"
ss120	1751080	1928	"marine, low-light"
S6803	3956956	3722	"fresh water"

C. Accessing data from data aggregates

Data is extracted from data aggregates with the [] operator. The square brackets mean "Look inside the object that precedes the brackets and get the item or items specified within the brackets (at present, it is possible to specify only one item to be extracted at one time from a table). The arrow (`->`) operator allows you to specify ranges of items. When you use the arrow operator, be sure to precede it and follow it with a space, otherwise BBL may combine it with the name of an adjacent symbol.

Work through the examples in the problem set that follows to see how brackets and arrows work.

Problem Set
Data Aggregates

A. Strings

A.1. From the example below:

```
(DEFINE test-string AS "ABCDEFG")
test-string[2 -> 4]
test-string[2 5 1]
```

complete the following to extract the third codon from the gene sequence:

```
(DEFINE gene-seq AS (SEQUENCE-OF ssr1600)
fill in
```

Confirm what you did by displaying the sequence and looking for the third codon.

A.2. Write code that will extract the numbers from a social security number of the form: NNN-NN-NNNN

A.3. The third position of codons is the position of greatest variability. It therefore may be that nucleotide frequencies vary more at that position than at position 1 and 2, and so it may be a better (or maybe a worse!) diagnostic for the genome identity of a sequence. Find the total nucleotide frequencies within the genes of *Prochlorococcus marinus* SS120 at each of the three positions. The following function might be of use to you:

```
(FROM 1 TO 30 BY 3)
  --> (1 4 7 10 13 16 19 22 25 28)
```

B. Lists

B.1. From the example below:

```
(DEFINE test-list AS {1 2 3 4 5})
test-list[2 -> 4]
test-list[2 5 1]
```

Define the set of cyanobacteria that live in the ocean. Here are all available cyanobacteria:

```
*all-organisms*
```

Of these, all *Prochlorococci*, *Synechococci*, *Crocophaera*, and *Trichodesmium* live in the ocean.

C. Tables

C.1. From the example below:

```
(DEFINE test-table[1 "A"] AS "1A")
(DEFINE test-table[2 "G"] AS "2G")
(DISPLAY-TABLE test-table)
```

Write a loop that creates and displays a multiplication table, showing the products of ***m*** times ***n*** for ***m*** and ***n*** going from 1 to 10.