2008 Cognitive Systems Programming Question

Congratulations! **60-Minute Software** ("Ready in an hour or it's free!") has hired you help build their new PHAT (Pure HTML Active Templates) web page generation system. Unlike JSP, ASP, PHP, etc., PHAT uses only pure HTML. Elements to be replaced with dynamic data are labeled using class="*name*" attributes. Critical to making this work is inserting lists of data into example HTML. For example, here 3 sample scores are replaced with the scores 10, 15, 20, 25, and 30:

Input HTML	Output HTML
`<p><span class="score">1</span> <span class="score">2</span> <span class="score">3</span></p>`	`<p><span class="score">10</span> <span class="score">15</span> <span class="score">20</span> <span class="score">25</span> <span class="score">30</span></p>`
`<p>Scores: <span class="score">1</span>, <span class="score">2</span> and <span class="score">3</span>.</p>`	`<p>Scores: <span class="score">10</span>, <span class="score">15</span>, <span class="score">20</span>, <span class="score">25</span> and <span class="score">30</span>.</p>`

`<table>`
`  <tr><th>Scores</th></tr>`
`  <tr><td class="score">1</td></tr>`
`  <tr><td class="score">2</td></tr>`
`  <tr><td class="score">3</td></tr>`
`</table>`
	`<table>`
	`  <tr><th>Scores</th></tr>`
	`  <tr><td class="score">10</td></tr>`
	`  <tr><td class="score">15</td></tr>`
	`  <tr><td class="score">20</td></tr>`
	`  <tr><td class="score">25</td></tr>`
	`  <tr><td class="score">30</td></tr>`
	`</table>`

The HTML must have exactly 3 sample values with the same name. The first and last will be replaced with the first and last real values, and the middle sample will be replicated as needed to hold the middle values. Note that the sample values may be nested in other HTML that must be replicated to produce the proper result.

Your assistants have already written code to parse an HTML document into a DOM node called `document`, and they will define for you:

- `getNodes(`*`node, name`*`)` -- `get-nodes` in Lisp -- to return a collection (whatever form you wish: array, vector, list, …) of DOM nodes under the given node with the given class name, e.g., the SPAN or TD elements above, and
- `fill(`*`node, text`*`)` to replace the text content of a node.

Using these functions and the standard W3C DOM API (see next page), define `fillNodes(`*`collection, strings, name`*`)` -- `fill-nodes` in Lisp -- to take that collection of nodes, a collection of strings, and the class name. It should replicate DOM nodes appropriately, and call `fill()` to replace their contents. <u>You can assume there are always at least 3 data values.</u>

W3C Document Object Model (DOM) API

The W3C DOM models an HTML document as a tree of Node objects, e.g., <body> Hello, world! </body> is a BODY node with one P child, the P child has one text node child, and the text node child contains the string “Hello, world!”. Here are the DOM functions you need, as documented for Java's `org.w3c.Node` class and Common Lisp's `dom-node` class. Analogous functions exist in C++ and other C-family languages.

```
Node appendChild(Node newChild)
dom-node dom-append-child (node, newChild)
```

Adds the node newChild to the end of the list of children of this node.

```
Node cloneNode(boolean deep)
dom-node dom-clone-node(node, deep)
```

Returns a duplicate of this node. If deep is true, recursively clones child nodes.

```
Node getFirstChild()
dom-node dom-first-child(node)
```

The first child of this node, or null (nil).

```
Node getLastChild()
dom-node dom-last-child(node)
```

The last child of this node, or null (nil).

```
Node getNextSibling()
dom-node dom-next-sibling(node)
```

The node immediately following this node, or null (nil).

```
Node getParentNode()
dom-node dom-parent-node(node)
```

The parent of this node, or null (nil).

```
Node getPreviousSibling()
dom-node dom-previous-sibling(node)
```

The node immediately preceding this node, or null (nil).

```
Node insertBefore(Node newChild, Node refChild)
dom-node dom-insert-before(node, new-child, ref-child)
```

Inserts (and returns) the node newChild before the existing child node refChild.