Purpose

To translate BridgeNet XML into a display of the hands and results in HTML format.

Rough Process (needs updating)

1) Read xml file as a string into a variable
2) Use xmlminidom to process the string into “node”s
3) Iterate through all board nodes
    1. Read vuln, dealer, and boardnum tags
    2. Create '<div class=”board”>'
    3. Add div class of “boardnum” with contents being boardnum
    4. Iterate through hands
        1. Create '<div>'s of class hand direction for CSS to position correctly.
        2. If direction is vuln then add '<div class=”vulnerable”>Vulnerable</div>'
        3. If direction is dealer then add '<div class=”dealer”>Dealer<div>'
        4. Iterate through Suites
            1. Create '<div>'s of class suite name with the card nums
    5. End “board” div
    6. Create table with class “results” to hold results
    7. Iterate through Result nodes
        1. Create table rows for each result
    8. End table of results
4) Output html header with style sheet, result from 3, and footer to file.

Data Structures (needs updating)

The only specialized data structures are for usage with xmlminidom:

```
(mv nodename attributes children) |  (mv 'text nil value)    ;node and text
attributes = (cons attribute attributes) | nil               ;attributes
attribute = (mv attributename value)                         ;attribute
nodes = (cons node nodes) | nil                              ;nodes
children = nodes                                             ;children
```

Interfaces and Contracts

xmlminidom:

xml-readnode (xmlchars) → returns the root node from xmlstring

Given any input xml-readnode will either return a structure of type xml-isnode or nil.

xml-getnodes (node nodename) → returns children of node with type nodename

xml-getdeepnodes (node nodename) → returns children of node with type nodename searching recursively using DFS with node as root.

Assuming (xml-isnode node) will return:

- something of type xml-isnodelist
- every node with name nodename.

xml-getnode (node nodename) → returns first child node with type nodename

xml-getdeepnode (node nodename) → returns first child node with type nodename searching recursively using DFS with node as root.

Assuming (xml-isnode node) will return:

- something of type xml-isnode
- every node with name nodename.

xml-getattribute (node attributename) → returns the value of node's attribute with name attributename

Assuming (xml-isnode node) will return a string (empty string if not found)

xml-gettext (node) → returns the composite of all text inside of a node

Assuming (xml-isnode node) will return a string (empty string if no 'text elements)

xml-isattribute (attribute) → returns true iff attribute is an mv of length 2 with both elements of the mv being strings

xml-isattributelist (attributes) → returns true iff attributes is nil or a list of mv's of length 2 with both elements of each mv being strings

xml-isnode (node) → returns true iff node is actually a node

xml-isnodelist (nodes) → returns true iff nodes is a list of nodes or nil