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

board:

getresults (xmlnodes prefix postfix) → serializes xmlnodes to HTML table with prefix and postfix appended before and after respecitvely

getseparateresults (xmlnodes) → serializes xmlnodes to a sequence of HTML tables corresponding to the seperate results for each player

getboards (xmlnodes) → Returns a serialization of the "board" class divs from xmlnodes, a list of xml nodes. The resulting serialization will look something like:

```
<div class="board">
<div class="boardnum">Board: 33</div>
(33 comes from the contents of the BoardNo element)
<div class="N">
(N comes from the direction attribute of the Hand element)
  <div class="dealer">Dealer</div>
  <div class="vulnerable">Vulnerable</div>
  &spades;234<br />
</div>
</div>
*tablehead*
```

<tr>
<td>A3</td>
(A3 comes from the SectionLabel + PairID-NS)
<td>A5</td>
(A5 comes from the SectionLabel + PairID-NS)
<td>5.0</td>
(5.0 comes from the contents of the TotalScore element)
<td>&nbsp;</td> (XXX why this?)
<td>120.0</td>
(120.0 comes from the contents of the MatchpointsNS element)
<td>120.0</td>
(120.0 comes from the contents of the MatchpointsEW element)
</tr>
**tabletail*)))*

gethands (xmlnodes vulnerable dealer) → serialize hands into divs, where the class is the hand direction derived from xmlnodes, the "vulnerable" and "dealer" divs are added inside as necessary, and the cards are added to each hand via gethandcards.
xmlnodes conforms to the minidom structure and is of type hand.
vulnerable is the text content of the Vulnerable node
dealer is the text content of the Dealer node

gethandcards (xmlnodes) → serialize the hand cards from xmlnodes where xmlnodes is a list of node structures representing "Suit"s.
For...
<Suit symbol="S">832</Suit>
<Suit symbol="H">QT42</Suit>
<Suit symbol="D">A865</Suit>
<Suit symbol="C">A9</Suit>
You will get something like
*&spades;832
*
*&hearts;QT42
*
*&diams;A865
*
*&clubs;A9
*

psc:

getPSC (xmlnodes) → Given xmlnodes, this returns a string PSC-HTML HTML table for each PSC
Notes: Recurses for every PairID, parsing out each pair's score card

getNameForID (pairid, data) → The string pairid needs to define the direction (E-W). section (A, B), and Number (1,2,3) returns string list (list NameOfPerson1 NameOfPerson2)
Notes: Pulls a pair's names from the nodes data

getBoardsForPair (pairid, results) → Given string pairid, where PairID needs to define the direction (E-W). section (A, B), and Number (1,2,3), and node results returns a string Boards-HTML which is a HTML table with rows for each match
Board, Direction, Versus, Score, Matchpoints are the columns
Notes: Uses getSeparateResults to pull one pair's results

rankings:

getRankings (rankings) → Given list of nodes rankings returns a string Rankings-HTML
HTML will be some header information then a table for each Section/Direction pair
Columns: Pair No., Players, Strat, Overall Rank (A, B, C), Section Rank (A,B,C), Matchpoint Score, Percentage Score, Masterpoint Award
Notes: Parses out the rankings data into html tables

getContestant (pairid, rankings) → Given string pairid, where PairID needs to define the direction (E-W) section (A, B) & Number (1,2,3), and rankings nodes returns a list of Strings, one string value for each of the following:
Pair No., Players, Strat, Overall Rank (A, B, C), Section Rank (A,B,C), Matchpoint Score, Percentage Score, Masterpoint Award
Notes: Pulls a single pair's results out of the rankings data