Purpose

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

Rough Process

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 for each of these generate HTML
    1. Make boards with travelers
    2. Make boards without travelers
    3. Make personal score card
    4. Make rankings
4) Output html headers with style sheets, results from 3, and footers to files.

Data Structures

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-serizlize-nodes (xmlnodes) → Returns a string containing xml nodes that represents the node list, xmlnodes.

xml-serizlize-dom (xmlnode) → Returns a string containing an xml document that represents the dom passed in through xmlnode.

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 descendants of node with type nodename, searching recursively using DFS with node as root.

Assuming (xml-isnode node) is true, xml-getdeepnodes 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) is true, xml-getdeepnode 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-bfsfindnodes (nodes nodename) → returns a list of children nodes of type nodename using BFS search that are at the shallowest depth.

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:

serializedresults (xmlnodes) → returns a string consisting of the concatenation of results table rows from each "Result" node

getseparateresults (xmlnodes boardnum ns1 ew1) → grabs the results for one board separately from the board information without html serialization where xmlnodes is the list of results for a board, boardnum is the boardnum, ns1 and ew1 are the initial lists. It returns a structure like:

```
(mv ns ew)
where
 ns = (list
        (cons (mv pairns section)
              (list
                (mv boardnum pairew totalscore pointsns))
                …)
        …)
 ew = (list
        (cons (mv pairew section)
              (list
                (mv boardnum pairns totalscore pointsns))
                …)
        …)
```

getallseparateresults (boardnodes) → converts boardnodes, a list of Board nodes, to a sequence like:

```
(mv ns ew)
where
 ns = (list
       (cons (mv pairns section)
             (list
               (mv boardnum pairew totalscore pointsns))
               …)
       …)
 ew = (list
       (cons (mv pairew section)
             (list
               (mv boardnum pairns totalscore pointsns))
               …)
       …)
```

serializedboards (xmlnodes) → returns appended "board" class divs with their "results" tables from the xmlnode "Board" and "results" formatted to be rendered with the deal and results as required by description.

serializehands (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 serializehandcards.
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

serializehandcards (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<br/>
&hearts;QT42<br/>
&diams;A865<br/>
&clubs;A9<br/>
```

psc:

getpsc (xmlnodes) → Given xmlnodes, this returns a string 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: Iterates over the list returned from getseperateresults to pull one pair's board results

rankings:

getrankings (rankingnodes) → Given list of ranking nodes, returns a string HTML will be some header information then a table for each Section/Direction pair
Columns: Pair No. (section, pair ID, and direction, e.g., "A1 N-S"); Players; Strat; an Overall Rank mv for Strats A, B; and C, Section Rank mv for Strats A, B, and C; Matchpoint Score; Percentage Score; Masterpoint Award
Notes: Parses out the rankings data into html tables

getcontestants (section dir id sections) → Find the Contestants node with ID id, in the Section in sections (a list of Section nodes), where Section has Direction dir and SectionLabel section.

getcontestantsnames (contestants) → For the Contestants node given by contestants, extract the names from the text contents of the Player nodes.

io:

main(bridgeXML state)→Given a duplicate bridge XML file, extracts appropriate information to creates four HTML pages that link together: boards, boards with travelers, rankings, and personal score card webpages.

boards-no-trav(bridgeXML state)→Given a duplicate bridge XML file, gets the following information from each board listed in the XML file and creates an HTML page with this information formatted into tables:
Board numbers, dealer, vulnerable, and hands for each direction

boards-trav(bridgeXML state)→ Given a duplicate bridge XML file, gets the same information as boards-no-trav as well as the travelers information for each board, which includes the total score and matchpoints for all pairs at that board.

Creates HTML page with this information with the board information and travelers information in separate tables.

rankings(bridgeXML state)→ Given a duplicate bridge XML file, creates a rankings table in an HTML file including each pairs ranking and various other stats such as matchpoint and percentage score.

personal-score-cards(bridgeXML state)→Given a duplicate bridge XML file, creates an HTML file containing personal score cards for each pair, which includes information about each match they played an against whom.

PROBE Software Size Estimate:

Reused Functions	*LOC*
xml-gettext	*9*
xml-getattribute	*8*
xml-getnode	*2*
xml-getnodes	*10*
xml-readnode	*5*
xml-readnodes	*53*
xml-skipdontcares	*20*
xml-readnodeproperties	*24*
xml-unescape	*27*
splitoff-prefix-mv	*3*
split-at-delimiter-mv	*3*
span-mv	*3*
split-on-token-mv	*3*
serializeboards	*32*
getresults	*35*
serializehands	*62*
serializehandcards	*32*
suit-list?	*8*
suit?	*38*

Total Reused LOC: 377

Historical Data Table for PROBE

	Tiny(7%)	*Small(24%)*	*Medium(38%)*	*Large(24%)*	*Huge(7%)*
Avg. LOC	*1.69*	*3.82*	*7.34*	*16.31*	*41.71*

Predicted Functions	*Predicted Size*
xml-getdeepnodes	*Medium*
serializeseparateresults	*Large*
serializeresults	*Huge*
getPSC	*Large*
getnameforID	*Medium*
getboardsforpair	*Medium*
getrankings	*Medium*
getcontestant	*Medium*
boards-no-trav	*Small*
boards-trav	*Small*
rankings	*Small*
personal-score-cards	*Medium*
main	*Large*

Total Predicted LOC: 146.14

Total Estimated LOC: 523