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
    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

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
```

Functions

iEx5:

main (infile outfile state) → Converts infile from bridgenet xml to html output and saves in outfile; returns (mv 'ok state) or (mv 'error state)

getboards (xmlnode) → 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

getresults (xmlnodes prefix postfix) → returns a string consisting of the concatenation of prefix, results table rows from each "Result" node, and postfix gethands (xmlnodes vulnerable dealer) → returns concatenated list of divs with class set to hand direction from xmlnodes, where xmlnodes is a list of xmlnode, of type hand, adds “vulnerable” and “dealer” divs inside the divs as necessary, and adds the cards to each hand

gethandcards (xmlnodes) → returns concatenated list of strings composed of the concatenation of suite symbol in HTML and card characters from xmlnodes where xmlnodes is a list of Suite xml nodes

suit? (xmlnode) → returns true if xmlnode is of the following form:

```
(mv "Suit"
  (list (mv "symbol" ("S"||"H"||"D"||"C")))
  (list (mv 'text nil stringp)))
```

suit-list? (xmlnodes) → returns true if suit? is true for each item in the list xmlnodes

stringutils:

stringlist-append (stringlist) → returns a string that is each string in stringlist appended together in order

stringlist? (stringlist)→ returns true if stringlist is a list of strings

```
#| Edward Flick
   Software Engineering I
   iEx5_test

   Tests for the iEx5 assignment.
|#

(in-package "ACL2")
(include-book "testing" :dir :teachpacks)
(include-book "doublecheck" :dir :teachpacks)
(include-book "iEx5")

; Removed all tests from iEx4 to keep this clean
; Only testing new functions

; getresults (xmlnodes prefix postfix)
(check-expect (getresults nil *tablehead* *tabletail*)
              (string-append *tablehead* *tabletail*))
(check-expect
 (getresults
  (list
   (mv
    "Result"
    (list
     (mv "SectionLabel" "A")
     (mv "PairID-NS" "3")
     (mv "PairID-EW" "5")
     )
    (list
     (mv "TotalScore"
         (list (mv "Direction" "N-S"))
         (list (mv 'text nil "5.0"))
         )
     (mv "MatchpointsNS" nil (list (mv 'text nil "120.0")))
     (mv "MatchpointsEW" nil (list (mv 'text nil "120.0")))
     )
   ))
  *tablehead*
  *tabletail*)
 (stringlist-append
  (list
   *tablehead*
   "<tr>"
   "<td>A3</td>"
   "<td>A5</td>"
   "<td>5.0</td><td>&nbsp;</td>"
   "<td>120.0</td>"
   "<td>120.0</td>"
   "</tr>"
   *tabletail*)))
(check-expect
 (getresults
  (list
   (mv
    "Result"
    (list
     (mv "SectionLabel" "A")
     (mv "PairID-NS" "3")
     (mv "PairID-EW" "5")
     )
    (list
     (mv "TotalScore"
         (list (mv "Direction" "E-W"))
         (list (mv 'text nil "7.0"))
         )
     (mv "MatchpointsNS" nil (list (mv 'text nil "120.0")))
     (mv "MatchpointsEW" nil (list (mv 'text nil "120.0")))
     )
   ))
  *tablehead*
  *tabletail*)
 (stringlist-append
  (list
   *tablehead*
   "<tr>"
   "<td>A3</td>"
   "<td>A5</td>"
   "<td>&nbsp;</td><td>7.0</td>"
   "<td>120.0</td>"
   "<td>120.0</td>"
   "</tr>"
   *tabletail*)))

; getboards (xmlnode)
(check-expect (getboards nil) "")
(check-expect
 (getboards
  (list
   (mv
    "Board"
    nil
    (list
     (mv "Vulnerable" nil (list (mv 'text nil "NS")))
     (mv "Dealer" nil (list (mv 'text nil "N")))
     (mv "BoardNo" nil (list (mv 'text nil "33")))
     (mv "Deal" nil
         (list
          (mv
           "Hand"
           (list (mv "direction" "N"))
           (list
            (mv "Suit"
                (list (mv "symbol" "S"))
                (list (mv 'text nil "234")))
            ))
          )))))
  )
 (stringlist-append (list
  "<div class=\"board\">"
  "<div class=\"boardnum\">Board: 33</div>\n"
  "<div class=\"N\">"
  "<div class=\"dealer\">Dealer</div>\n"
  "<div class=\"vulnerable\">Vulnerable</div>\n"
  "&spades;234<br />\n"
  "</div>\n"
  "</div>\n"
  *tablehead*
  *tabletail*)))
(check-expect
 (getboards
  (list
   (mv
    "Board"
    nil
    (list
     (mv "Vulnerable" nil (list (mv 'text nil "NS")))
     (mv "Dealer" nil (list (mv 'text nil "N")))
     (mv "BoardNo" nil (list (mv 'text nil "33")))
     (mv "Deal" nil
         (list
          (mv
           "Hand"
           (list (mv "direction" "N"))
           (list
            (mv "Suit"
                (list (mv "symbol" "S"))
                (list (mv 'text nil "234")))
            ))
          ))
     (mv
      "Result"
      (list
       (mv "SectionLabel" "A")
       (mv "PairID-NS" "3")
       (mv "PairID-EW" "5")
       )
      (list
       (mv "TotalScore"
           (list (mv "Direction" "N-S"))
           (list (mv 'text nil "5.0"))
           )
       (mv "MatchpointsNS" nil (list (mv 'text nil "120.0")))
       (mv "MatchpointsEW" nil (list (mv 'text nil "120.0")))
       )
      )
     )))
  )
 (stringlist-append (list
  "<div class=\"board\">"
  "<div class=\"boardnum\">Board: 33</div>\n"
  "<div class=\"N\">"
  "<div class=\"dealer\">Dealer</div>\n"
  "<div class=\"vulnerable\">Vulnerable</div>\n"
  "&spades;234<br />\n"
  "</div>\n"
  "</div>\n"
  *tablehead*
  "<tr>"
  "<td>A3</td>"
  "<td>A5</td>"
  "<td>5.0</td><td>&nbsp;</td>"
  "<td>120.0</td>"
  "<td>120.0</td>"
  "</tr>"
  *tabletail*)))
```

```
#| Edward Flick
   Software Engineering I
   iEx5_pred

   Predicate tests for the iEx5 assignment.
|#

(in-package "ACL2")
(include-book "testing" :dir :teachpacks)
(include-book "doublecheck" :dir :teachpacks)
(include-book "iEx5")

; Removed all tests from iEx4 to keep this clean
; Only testing new functions

(defproperty getresults-nil=string-append-prefix-postfix-tst :repeat 100
  (prefix :value (random-string)
   postfix :value (random-string))
  (implies (and (stringp prefix) (stringp postfix))
           (string-equal (getresults nil prefix postfix)
                         (string-append prefix postfix)))
)

(defproperty getresults-results-proportional-to-input-tst :repeat 100
  (n     :value (random-between 1 200)
   nodes :value (random-list-of
                 (mv
                  "Result"
                  (list
                   (mv "SectionLabel" "A")
                   (mv "PairID-NS" "3")
                   (mv "PairID-EW" "5")
                   )
                  (list
                   (mv "TotalScore"
                       (list (mv "Direction" "N-S"))
                       (list (mv 'text nil "5.0"))
                       )
                   (mv "MatchpointsNS" nil (list (mv 'text nil "120.0")))
                   (mv "MatchpointsEW" nil (list (mv 'text nil "120.0")))
                   )
                  )
                 :size n))
  (implies (> n 0)
           (< (length (getresults (cdr nodes) "" ""))
              (length (getresults nodes "" ""))))
)
```

```
#| Edward Flick
   Software Engineering I
   iEx5_ver

   Theorems for the iEx5 assignment.
|#

(in-package "ACL2")
(include-book "testing" :dir :teachpacks)
(include-book "doublecheck" :dir :teachpacks)
(include-book "iEx5")

; No theorems execute correctly becuase xmlminidom does not admit
(defthm getresults-nil=string-append-prefix-postfix-thm
  (implies (and (stringp prefix) (stringp postfix))
           (string-equal (getresults nil prefix postfix)
                         (string-append prefix postfix))))

(defthm getresults-nil-returns-a-string-thm
  (implies (and (stringp prefix) (stringp postfix))
           (stringp (getresults nil prefix postfix))))
```

```
name: Edward Flick
date: Oct 28, 2010
program: iEx5
instructor: Dr. Page
language: ACL2
actual added lines: 53
actual base lines: 394
actual modified lines: 42
actual removed lines: 0

time log:

        - date: Oct 21, 2010
          start time: 12:00PM
          end time: 12:30PM
          phase: design
          comment: Drafted basic design document. Included necessary functions, data types and rough
process.

        - date: Oct 21, 2010
          start time: 12:30PM
          end time: 2:30PM
          phase: code
          comment: Implemented test suite and wrote this file.

        - date: Oct 28, 2010
          start time: 10:00PM
          end time: 10:30PM
          phase: design
          comment: Updated the design, removed an unnecessary defun

    - date: Oct 28, 2010
          start time: 10:30PM
          end time: 11:50PM
          phase: code
          comment: Wrote the totality of the implementation.

        - date: Oct 29, 2010
          start time: 5:00PM
          end time: 8:30PM
          phase: code
          comment: Finished all the tests, predicates, and theorems.


new objects:
        - name: getboards
          type: Calculation
          estimated lines: 21

        - name: getresults
          type: Calculation
          estimated lines: 12

defect log:

        - date: Oct 21, 2010
          type: code
          fix time: 1
          comment: Don't know. Can't run tests without implementations.

        - date: Oct 28, 2010
          type: design
          fix time: 1
          comment: Removed an unnecessary defun.

        - date: Oct 28, 2010
          type: code
          fix time: 1
          comment: Forget to change recursive call of getgames to getboards.

        - date: Oct 28, 2010
          type: code
          fix time: 1
          comment: Updated CSS to format things more legibly.

        - date: Oct 28, 2010
          type: code
          fix time: 1
          comment: Updated tests to more thoroughly cover the new functions.

        - date: Oct 29, 2010
          type: code
          fix time: 1
          comment: Forgot to limit count of list in predicate test to make it practical.
```

iEx5

Personal Software Process Summary

Project Essentials

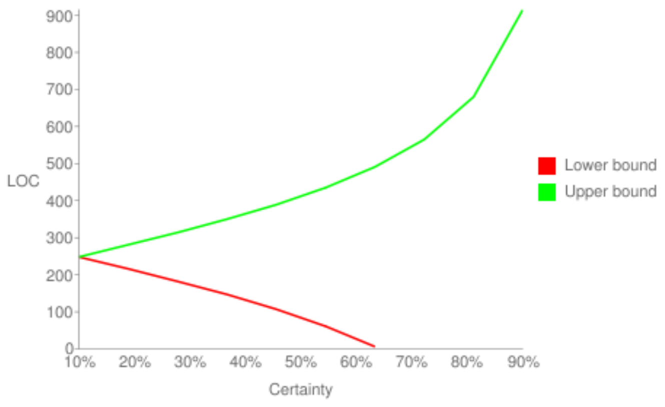
Name:	Edward Flick
Instructor:	Dr. Page
Date:	Oct 28, 2010
Language:	ACL2

Lines of Code

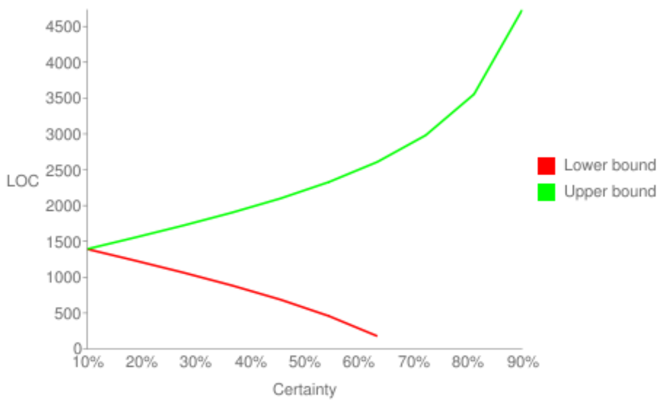
Type	Prediction by user	Actual
Added	33	53
Base	0	394
Modified	0	42
Removed	0	0

PSP Projection

LoC Certainty

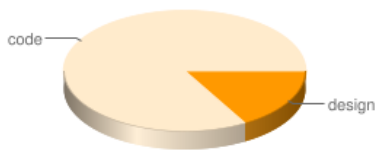


Time Certainty

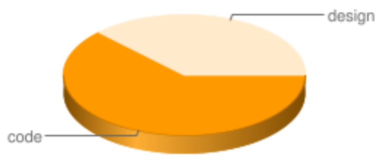


Project Data

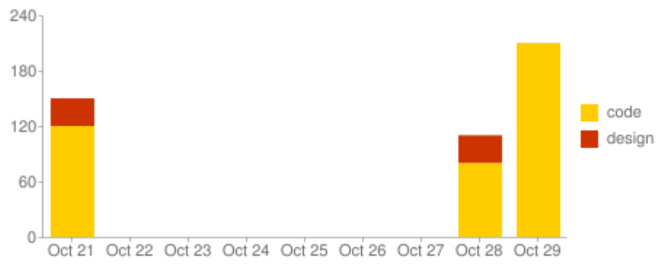
Time Per Defect Type



Time Per Phase

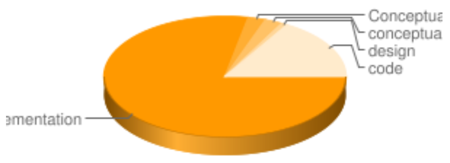


Time by Day

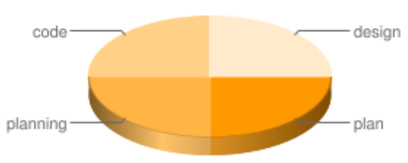


Cumulative Data

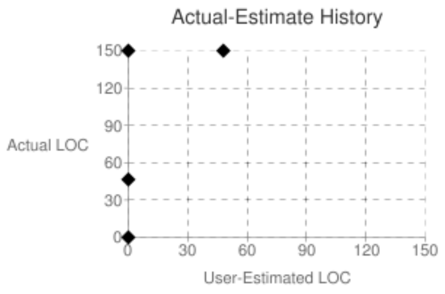
Time Per Defect Type



Time Per Phase



Actual vs Estimated LoC



Actual vs Estimated LoC

Project	Estimate	Actual
iEx4	32	244
iEx3	0	290
iEx2	0	0
iEx1	0	31

Time Log

Date	Type	Int. Time	Description
Oct 21, 2010, 12:00 PM - 12:30 PM	design	0	Drafted basic design document. Included necessary functions, data types and rough process.
Oct 21, 2010, 12:30 PM - 2:30 PM	code	0	Implemented test suite and wrote this file.
Oct 28, 2010, 10:00 PM - 10:30 PM	design	0	Updated the design, removed an unnecessary defun
Oct 28, 2010, 10:30 PM - 11:50 PM	code	0	Wrote the totality of the implementation.
Oct 29, 2010, 5:00 PM - 8:30 PM	code	0	Finished all the tests, predicates, and theorems.

Defect Log

Date	Phase	Fix Time	Description
Oct 21, 2010	code	1	Don't know. Can't run tests without implementations.
Oct 28, 2010	design	1	Removed an unnecessary defun.

Oct 28, 2010	code	1	Forget to change recursive call of getgames to getboards.
Oct 28, 2010	code	1	Updated CSS to format things more legibly.
Oct 28, 2010	code	1	Updated tests to more thoroughly cover the new functions.
Oct 29, 2010	code	1	Forgot to limit count of list in predicate test to make it practical.

```
#| Edward Flick
   Software Engineering I
   iEx5

   The iEx5 assignment to format bridge hands as HTML.
|#

(in-package "ACL2")

(include-book "io-utilities" :dir :teachpacks)
;(include-book "basiclex")
(include-book "xmlminidom")
(include-book "stringutils")

(set-state-ok t)

(defconst *div-open-1* "<div class=\"")
(defconst *div-open-2* "\">")
(defconst *div-close* "</div>\n")
(defconst *br* "<br />\n")
(defconst *htmlhead*
  (stringlist-append
   (list
    "<html><head><style>"
    "body {background-color: white; color: black;}"
    ".board {clear: both; position: relative; top: 0px; left: 0px;"
            "width: 45em; height: 30em; border: none;"
            "margin: 1em 1em 0em; background-color: #f8f8f8;}\n"
    ".boardnum {position: absolute; left: 0em; top: 0em;}\n"
    ".N {position: absolute; height: 10em; width: 15em; left: 15em;"
        "top: 0em; border: dashed 1px black; background-color: white;}\n"
    ".S {position: absolute; height: 10em; width: 15em; left: 15em;"
        "top: 20em; border: dashed 1px black; background-color: white;}\n"
    ".E {position: absolute; height: 10em; width: 15em; left: 30em;"
        "top: 10em; border: dashed 1px black; background-color: white;}\n"
    ".W {position: absolute; height: 10em; width: 15em; left: 0em;"
        "top: 10em; border: dashed 1px black; background-color: white;}\n"
    ".dealer {text-align: center; font-weight: bold; color: blue;}\n"
    ".vulnerable {text-align: center; font-weight: bold; color: red;}\n"
    ".results {clear: none; float: left; width: 45em;"
              "margin: 0em 1em 1em;}\n"
    ".results tr * {border: 1px solid black; margin: 1px;}\n"
    "</style></head><body>")))
(defconst *htmltail* "</body></html>")
(defconst *tablehead*
  (stringlist-append
   (list
    "<table class=\"results\"><tr>"
    "<th colspan=\"2\">Pairs</th>"
    "<th colspan=\"2\">Total Score</th>"
    "<th colspan=\"2\">Match Points</th>"
    "</tr><tr>"
    "<th>NS</th>"
    "<th>EW</th>"
    "<th>NS</th>"
    "<th>EW</th>"
    "<th>NS</th>"
    "<th>EW</th>"
    "</tr>"
    )))
(defconst *tabletail* "</table>\n")

; suit? (xmlnode) → returns true if xmlnode is of the following form:
; (mv "Suit"
;     (list (mv "symbol" ("S"||"H"||"D"||"C")))
;     (list (mv 'text nil stringp)))
(defun suit? (xmlnode)
  (and
   (true-listp xmlnode)
   (equal (len xmlnode) 3)
   (mv-let
    (node attribs children)
    xmlnode
    (and
     (equal node "Suit")
     (true-listp attribs)
     (equal (len attribs) 1)
     (let
         ((a (car attribs)))
       (and
        (true-listp a)
        (equal (len a) 2)
        (mv-let
         (an av)
         a
         (and
          (equal an "symbol")
          (member-equal av (list "S" "H" "D" "C"))
          ))))
     (true-listp children)
     (equal (len children) 1)
     (let
         ((c (car children)))
       (and
        (true-listp c)
        (equal (len c) 3)
        (mv-let
         (nodetype dontcare contents)
         c
         (and
          (equal nodetype 'text)
          (null dontcare)
          (stringp contents)
          ))))))))

; suit-list? (xmlnodes) → returns true if suit? is true for each item in
; the list xmlnodes
(defun suit-list? (xmlnodes)
  (and
   (true-listp xmlnodes)
   (or
    (null xmlnodes)
    (and
     (suit? (car xmlnodes))
     (suit-list? (cdr xmlnodes))))))

; gethandcards (xmlnodes) → returns concatenated list of strings composed
; of the concatenation of suite symbol in HTML and card characters from
; xmlnodes where xmlnodes is a list of Suite xml nodes
(defun gethandcards (xmlnodes)
  (if (null xmlnodes)
      ""
      (let* (
             (suite
              (car xmlnodes))
             (rest
              (cdr xmlnodes))
             (suitesymbol
              (xml-getattribute suite "symbol"))
             (suitehtml
              (if
               (string-equal suitesymbol "S")
               "&spades;"
               (if
                (string-equal suitesymbol "C")
                "&clubs;"
                (if
                 (string-equal suitesymbol "D")
                 "&diams;"
                 "&hearts;"))))
             (cards
              (xml-gettext suite))
             )
        (stringlist-append
         (list
          suitehtml
          cards
          *br*
          (gethandcards rest)
          ))
        )))

; gethands (xmlnodes vulnerable dealer) → returns concatenated list of divs
; with class set to hand direction from xmlnodes, where xmlnodes is a list
; of xmlnode, of type hand, adds “vulnerable” and “dealer” divs inside the
; divs as necessary, and adds the cards to each hand
(defun gethands (xmlnodes vulnerable dealer)
  (if (null xmlnodes)
      ""
      (let* (
             (hand
              (car xmlnodes))
             (rest
              (cdr xmlnodes))
             (direction
              (xml-getattribute hand "direction"))
             (suites
              (xml-getnodes hand "Suit"))
             (dealerhtml
              (if (string-equal dealer direction)
                  (stringlist-append
                   (list
                    *div-open-1*
                    "dealer"
                    *div-open-2*
                    "Dealer"
                    *div-close*
                    ))
                  ""
               ))
             (vulnerablehtml
              (if
               (or
                (string-equal vulnerable "Both")
                (and
                 (string-equal vulnerable "NS")
                 (or
                  (string-equal direction "N")
                  (string-equal direction "S")
                 ))
                (and
                 (string-equal vulnerable "EW")
                 (or
                  (string-equal direction "E")
                  (string-equal direction "W")
                 ))
                )
               (stringlist-append
                (list
                 *div-open-1*
                 "vulnerable"
                 *div-open-2*
                 "Vulnerable"
                 *div-close*
                 ))
               ""
               ))
             )
        (stringlist-append
         (list
          *div-open-1*
          direction
          *div-open-2*
          dealerhtml
          vulnerablehtml
          (gethandcards suites)
          *div-close*
          (gethands rest vulnerable dealer))))))

;getresults (xmlnodes prefix postfix) → returns a string consisting of
; the concatenation of prefix, results table rows from each “Result” node,
; and postfix
(defun getresults (xmlnodes prefix postfix)
  (stringlist-append
   (list
    prefix
    (if (null xmlnodes)
        ""
        (let*
            (
             (result (car xmlnodes))
             (rest (cdr xmlnodes))
             (section (xml-getattribute result "SectionLabel"))
             (pairns (xml-getattribute result "PairID-NS"))
             (pairew (xml-getattribute result "PairID-EW"))
             (totalscorenode (xml-getnode result "TotalScore"))
             (totaldir (xml-getattribute totalscorenode "direction"))
             (totalscore (xml-gettext totalscorenode))
             (pointsns (xml-gettext (xml-getnode result "MatchpointsNS")))
             (pointsew (xml-gettext (xml-getnode result "MatchpointsEW")))
             )
          (stringlist-append
           (list
            "<tr>"
            "<td>" section pairns "</td>"
            "<td>" section pairew "</td>"
            "<td>" (if (string-equal totaldir "N-S")
                       totalscore "&nbsp;") "</td>"
            "<td>" (if (string-equal totaldir "E-W")
                       totalscore "&nbsp;") "</td>"
            "<td>" pointsns "</td>"
            "<td>" pointsew "</td>"
            "</tr>"
            (getresults rest "" "")
            ))))
    postfix
    )))

;getboards (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
(defun getboards (xmlnodes)
  (if (null xmlnodes)
      ""
      (let* (
             (game (car xmlnodes))
             (rest (cdr xmlnodes))
             (vulnerable
              (xml-gettext (xml-getnode game "Vulnerable")))
             (dealer
              (xml-gettext (xml-getnode game "Dealer")))
             (boardnum
              (xml-gettext (xml-getnode game "BoardNo")))
             (hands
              (xml-getnodes (xml-getnode game "Deal") "Hand"))
             (results
              (xml-getnodes game "Result"))
             )
        (stringlist-append
         (list
          *div-open-1*
          "board"
          *div-open-2*
          *div-open-1*
          "boardnum"
          *div-open-2*
          "Board: "
          boardnum
          *div-close*
          (gethands hands vulnerable dealer)
          *div-close*
          (getresults results *tablehead* *tabletail*)
          (getboards rest))))))


; main (infile outfile state) → Converts infile from bridgenet xml to html
; output and saves in outfile; returns (mv 'ok state) or (mv 'error state)
(defun main (infile outfile state)
  (mv-let (contents status state)
          (file->string (string-append infile ".xml") state)
          (if (null status)
              (mv-let
               (status state)
               (string-list->file
                (string-append outfile ".htm")
                (list
                 *htmlhead*
                 (getboards
                  (xml-getnodes (xml-getnode (xml-getnode
                   (xml-readnode contents)
                   "Game") "HandRecords") "Board"))
                 *htmltail*
                 )
                state)
               (if (null status)
                   (mv 'ok state)
                   (mv 'error state)))
              (mv 'error state))))

;(main "051115A" "testout1" state)
;(main "090303A" "testout2" state)
```