```
;; The first four lines of this file were added by Dracula.
;; They tell DrScheme that this is a Dracula Modular ACL2 program.
;; Leave these lines unchanged so that DrScheme can properly load this file.
#reader(planet "reader.ss" ("cce" "dracula.plt") "modular" "lang")
#| Team Steele
   Software Engineering I
   mBoard

   Software that creates a document object model from XML input.
|#

(require "../interfaces/Iboard.lisp")
(require "../interfaces/Ibasiclex.lisp")
(require "../interfaces/Ixmlminidom.lisp")
(module Mboard

  (import Ibasiclex)
  (import Ixmlminidom)

  (include-book "io-utilities" :dir :teachpacks)
  (include-book "list-utilities" :dir :teachpacks)

  ; serializedhandcards (xmlnodes) → returns concatenated list of strings
  ; composed of the concatenation of suit symbol in HTML and card
  ; characters from xmlnodes where xmlnodes is a list of Suit xml nodes
  (defun serializedhandcards (xmlnodes)
    (if (null xmlnodes)
        ""
        (let* ((suit (car xmlnodes))
               (rest (cdr xmlnodes))
               (suitsymbol (xml-getattribute suit "symbol"))
               (suithtml (if (string-equal suitsymbol "S")
                             "&spades;"
                             (if (string-equal suitsymbol "C")
                                 "&clubs;"
                                 (if (string-equal suitsymbol "D")
                                     "&diams;"
                                     "&hearts;"))))
               (cards (xml-gettext suit)))
          (concatenate 'string
                       suithtml
                       cards *br*
                       (serializedhandcards rest)))))

  ; serializedhands (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 serializedhands (xmlnodes vulnerable dealer)
    (if (null xmlnodes)
        ""
        (let* ((hand (car xmlnodes))
               (rest (cdr xmlnodes))
               (direction (xml-getattribute hand "direction"))
               (suits (xml-getnodes hand "Suit"))
               (dealerhtml (if (string-equal dealer direction)
                               (concatenate 'string
                                            *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"))))
                                   (concatenate 'string
                                                *div-open-1*
                                                "vulnerable"
                                                *div-open-2*
                                                "Vulnerable"
                                                *div-close*)
                                   "")))
          ; For your reading pleasure, this is the let* body
          (concatenate 'string
                       *div-open-1*
                       direction
                       *div-open-2*
                       dealerhtml
                       vulnerablehtml
                       (serializedhandcards suits)
                       *div-close*
                       (serializedhands rest vulnerable dealer)))))

  ; 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))
  ;                 ...)
  ;         ...)
  (defun getseparateresults (xmlnodes boardnum ns1 ew1)
    (if (null xmlnodes)
        (mv ns1 ew1)
        (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"))))
          (mv-let (ns ew)
                  (getseparateresults rest boardnum ns1 ew1)
            (let ((nskv (assoc-equal (mv pairns section) ns))
                  (ewkv (assoc-equal (mv pairew section) ew)))
                  (mv (cons (cons (mv pairns section)
                                  (cons (mv boardnum
                                            pairew
                                            (if (string-equal totaldir
                                                              "N-S")
                                                totalscore
                                               (string-append "-"
                                                              totalscore))
                                            pointsns)
                                        (cdr nskv)))
                            (remove-equal nskv ns))
                      (cons (cons (mv pairew section)
                                  (cons (mv boardnum
                                            pairns
                                            (if (string-equal totaldir
                                                              "E-W")
                                                totalscore
                                                (string-append "-"
                                                               totalscore))
                                            pointsew)
                                         (cdr ewkv)))
                            (remove-equal ewkv ew))))))))

  ; 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))
  ;                 ...)
  ;         ...)
  (defun getallseparateresults (boardnodes)
    (if (null boardnodes)
        (mv nil nil)
        (let* ((board (car boardnodes))
               (rest (cdr boardnodes))
               (boardnum (xml-gettext (xml-getnode board "BoardNo")))
               (results (xml-getnodes board "Result")))
          (mv-let (ns ew)
                  (getallseparateresults rest)
            (getseparateresults results boardnum ns ew)))))

  ; serializedresults (xmlnodes) → returns a string consisting of
  ; the concatenation of results table rows from each “Result” node
  (defun serializedresults (xmlnodes)
      (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"))))
            (concatenate 'string
                         "<tr>"
                         "<td><a href=\"psc.htm#N-S" pairns section "\">"
                         section pairns "</a></td>"
                         "<td><a href=\"psc.htm#E-W" pairew section "\">"
                         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>"
                         (serializedresults rest)))))

  ; 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
  (defun serializedboards (xmlnodes trav-flag)
    (if (null xmlnodes)
        ""
        (let* ((board (car xmlnodes))
               (rest (cdr xmlnodes))
               (vulnerable (xml-gettext (xml-getnode board "Vulnerable")))
               (dealer (xml-gettext (xml-getnode board "Dealer")))
               (boardnum (xml-gettext (xml-getnode board "BoardNo")))
               (hands (xml-getnodes (xml-getnode board "Deal") "Hand"))
               (results (xml-getnodes board "Result")))
          (concatenate 'string
                       *div-open-1*
                       "board\" id=\"" boardnum
                       *div-open-2*
                       *div-open-1*
                       "boardnum"
                       *div-open-2*
                       "Board: " boardnum "<br/><a href=\""
                       (if (equal trav-flag 1)
                           (concatenate
                            'string
                            "boards-no-trav.htm#"
                            boardnum "\"><small>(hide travelers)</small></a>")
                           (concatenate
                            'string
                            "boards-trav.htm#"
                            boardnum "\"><small>(show travelers)</small></a>")
                           )
                       *div-close*
                       (serializedhands hands vulnerable dealer)
                       *div-close*
                       (if (equal trav-flag 1)
                           (concatenate 'string
                                        *tablehead*
                                        (serializedresults results)
                                        *tabletail*)
                           "")
                       (serializedboards rest trav-flag)))))

  (defun getgamestring (gamenode)
    (let* ((clubgame (xml-gettext (xml-getnode gamenode "ClubGame")))
           (eventname (xml-gettext (xml-getnode (xml-getnode gamenode "Event")
                                                "EventName")))
           (session (xml-gettext (xml-getnode gamenode "Session")))
           (date (xml-gettext (xml-getnode gamenode "Date"))))
      (concatenate 'string
                   clubgame " " eventname ", " session ", " date)))
  (export Iboard))
```