```
#| 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"
```

```lisp
                 *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)
```