```
;; 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")
(require "../interfaces/Iio.lisp")
(require "../interfaces/Iboard.lisp")
(require "../interfaces/Ipsc.lisp")
(require "../interfaces/Irankings.lisp")
(require "../interfaces/Ixmlminidom.lisp")

(module Mio
  (import Ixmlminidom)
  (import Iboard)
  (import Irankings)
  (import Ipsc)

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

  ; boards-no-trav (bntXML 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
  (defun boards-no-trav (bntXML state)
    (mv-let (status state)
            (string-list->file (string-append "boards-no-trav" ".htm")
                               (list *htmlhd1*
                                     "Boards without Travelers"
                                     *htmlhd2*
                                     *menu*
                                     (serializedboards
                                       (xml-getnodes (xml-getnode
                                                       (xml-getnode bntXML
                                                         "Game")
                                                       "HandRecords")
                                                     "Board")
                                       0)
                                     *htmltail*)
                               state)
      (if (null status)
          (mv 'ok state)
          (mv 'error state))))

  ; boards-trav (btXML 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.
  (defun boards-trav (btXML state)
    (let* ((boardnode (xml-getnodes (xml-getnode (xml-getnode btXML "Game")
                                                 "HandRecords")
                                    "Board")))
      (mv-let (status state)
              (string-list->file (string-append "boards-trav" ".htm")
                                 (list *htmlhd1*
                                       "Boards with Travelers"
                                       *htmlhd2*
                                       *menu*
                                       (serializedboards boardnode 1)
                                       *htmltail*)
                                 state)
        (if (null status)
            (mv 'ok state)
            (mv 'error state)))))

  ; rankings (rnkXML 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.
```

```
(defun rankings (rnkXML state)
  (let* ((games (xml-getnode rnkXML "Game"))
         (sections (xml-getnodes games "Section")))
    (mv-let (status state)
            (string-list->file (string-append "rnk" ".htm")
                               (list *htmlhd1*
                                     "Rankings"
                                     *htmlhd2*
                                     *menu*
                                     (serializedRankingsHeader games)
                                     (serializedRankings sections)
                                     *htmltail*)
                               state)
      (if (null status)
        (mv 'ok state)
        (mv 'error state)))))


; personal-score-cards (pscXML 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.
(defun personal-score-cards (pscXML state)
  (let* ((gamenode (xml-getnode pscXML "Game"))
         (boardnodes (xml-getnodes (xml-getnode gamenode "HandRecords")
                                   "Board")))
    (mv-let (status state)
            (string-list->file (string-append "psc" ".htm")
                               (list *htmlhd1*
                                     "Personal Score Cards"
                                     *htmlhd2*
                                     *menu*
                                     (serializedPSC gamenode boardnodes)
                                     *htmltail*)
                               state)
      (if (null status)
        (mv 'ok state)
        (mv 'error state)))))

; main (bridgeXML state) Given a duplicate bridge XML file, extracts
; appropriate information to create four HTML pages that link together:
; boards, boards with travelers, rankings, and personal score card
; webpages.
(defun main (bridgeXML state)
  (mv-let (contents status state)
        (file->string (string-append bridgeXML ".xml") state)
        (if (null status)
            (let* ((xml (xml-readnode contents))
                   (bt (boards-trav xml state))
                   (bnt (boards-no-trav xml (cadr bt)))
                   (rnk (rankings xml (cadr bnt)))
                   (psc (personal-score-cards xml (cadr rnk))))
              (mv 'ok (cadr psc)))
            (mv 'error state))))

(export Iio))
```