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

(module Mrankings
  (import Ixmlminidom)

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

  ; Gets the matchpointtotal for a contestant as a rational
  (defun getmatchpointtotal (node)
    (str->rat (xml-gettext (xml-getnode node "MatchPointTotal"))))

  ; Insert function adapted from *site*
  (defun insert-sorted (a lst)
    (if (or (endp lst)
            (>= (getmatchpointtotal a)
                (getmatchpointtotal (car lst))))
        (cons a lst)
        (cons (car lst) (insert-sorted a (cdr lst)))))

  ; This function sorts the contestants by MatchPointTotal
  (defun sortcontestants (unsortedcontestants)
    (if (null unsortedcontestants)
        nil
        (let* ((contestant (car unsortedcontestants))
               (rest (cdr unsortedcontestants)))
          (insert-sorted contestant (sortcontestants rest)))))

  (defun sortmvranks (ranks mvs)
    (if (null ranks)
        mvs
        (let ((rank (car ranks))
              (rest (cdr ranks)))
          (mv-let (a b c)
                  mvs
                  (sortmvranks
                   rest
                   (mv
                    (if (equal (xml-getattribute rank "Strat") "A")
                        (xml-gettext rank)
                        a)
                    (if (equal (xml-getattribute rank "Strat") "B")
                        (xml-gettext rank)
                        b)
                    (if (equal (xml-getattribute rank "Strat") "C")
                        (xml-gettext rank)
                        c)))))))

  (defun serializedcontestants (contestants section direction)
    (if (null contestants)
        ""
        (let* ((contestant (car contestants))
               (rest (cdr contestants))
               (pairno (xml-getattribute contestant "ID"))
               (players (xml-getnodes contestant "Player"))
               (player1 (xml-gettext (xml-getnode (car players) "Name")))
               (player2 (xml-gettext (xml-getnode (cadr players) "Name")))
               (strat (xml-getattribute contestant "Strat"))
               (sectionranks (xml-getnodes contestant "SectionRank"))
               (overallranks (xml-getnodes contestant "OverallRank"))
               (matchpoint (xml-gettext
                             (xml-getnode contestant "MatchpointTotal")))
               (percentage (xml-gettext
```

```lisp
                            (xml-getnode contestant "Percentage")))
              (mpvalue (xml-gettext (xml-getnode contestant "Award")))
              (masterpoint (if (equal mpvalue "")
                               " "
                               mpvalue)))
        (concatenate 'string
                     "<tr>"
                     "<td><a href=\"psc.htm#" direction pairno section
                     "\">" pairno "</a></td>"
                     "<td>" player1 "<br />" player2 "</td>"
                     "<td>" strat "</td>"
                     (mv-let (a b c)
                             (sortmvranks sectionranks
                                          (mv " " " "
                                              " " ))
                             (concatenate 'string
                                          "<td>" a "</td>"
                                          "<td>" b "</td>"
                                          "<td>" c "</td>"))
                     (mv-let (a b c)
                             (sortmvranks overallranks
                                          (mv " " " " " " ))
                             (concatenate 'string
                                          "<td>" a "</td>"
                                          "<td>" b "</td>"
                                          "<td>" c "</td>"))
                     "<td>" matchpoint "</td>"
                     "<td>" percentage "</td>"
                     "<td>" masterpoint "</td>"
                     "</tr>"
                     (serializedcontestants rest section direction)))))

; XXX rankingnodes is a bad misnomer.  rankingnodes should definitely
; *not* be a list of Rankings nodes.  At a minimum, we need the Section
; nodes, too.
(defun serializedrankings (rankingnodes)
  (if (null rankingnodes)
      ""
      (let* ((ranking (car rankingnodes))
             (rest (cdr rankingnodes))
             (section (xml-getattribute
                       ranking
                       "SectionLabel"))
             (direction (xml-getattribute
                         ranking
                         "Direction"))
             (unsortedcontestants (xml-getnodes
                                   (xml-getnode ranking "Rankings")
                                   "Contestants"))
             (contestanthtml (serializedcontestants
                              (sortcontestants unsortedcontestants)
                              section direction)))
        (concatenate 'string
                     *rktablehead*
                     "<p align=\"CENTER\">"
                     "Section "
                     section
                     "  -- "
                     direction
                     "</p>"
                     contestanthtml
                     *rktabletail*
                     (serializedrankings rest)))))

  ;Pulls header information from the game node
  (defun serializedRankingsHeader (gamenodes)
    (let* ((Date (xml-gettext (xml-getnode gamenodes "Date")))
```

```lisp
               (Club (xml-gettext (xml-getnode gamenodes "ClubGame")))
               (Event (xml-gettext (xml-getnode
                                     (xml-getnode gamenodes "Event")
                                     "EventName")))))
          (concatenate 'string
                       "<h4 align=\"CENTER\">"
                       "Rankings - "
                       Date
                       "<br />"
                       Club
                       " - "
                       Event
                       "</h4>")))
; sectionnodes should be a list of Section nodes
(defun findspecificsection (sectionnodes label dir)
  (if sectionnodes
      ; linear search!
      (let* ((current (car sectionnodes)))
        (if (and (equal (xml-getattribute current "SectionLabel")
                        label)
                 (equal (xml-getattribute current "Direction")
                        dir))
            current
            (findspecificsection (cdr sectionnodes) label dir)))
      nil))

; nodes should be a list of Contestants nodes
(defun findspecificcontestants (nodes id)
  (if nodes
      (let* ((current (car nodes)))
        (if (equal (xml-getattribute current "ID") id)
            current
            (findspecificcontestants (cdr nodes) id)))
      nil))

(defun getcontestants (sectionlabel dir id sections)
  (let* ((section (findspecificsection sections sectionlabel dir)))
    (findspecificcontestants
                       (xml-bfsfindnodes (list section) "Contestants")
                       id)))

; For the two players in a Contestants element, delivers a string in the
; form "Alice - Bob".
; Contestants is the Contestants node
(defun getcontestantsnames (contestants)
  (let* ((players (xml-getnodes contestants "Player")))
    (concatenate 'string (xml-gettext (car players))
                 " - "
                 (xml-gettext (cadr players)))))

(defun getranks (ranksoftype)
  (if (< 0 (len ranksoftype))
    (let* ((rank (car ranksoftype)))
      (concatenate 'string
                   (xml-gettext rank)
                   "("
                   (xml-getattribute rank "Strat")
                   ")"
                   (getranks (cdr ranksoftype))))

    ""))

(defun getrankstring (ranktype contestants)
  (let* ((ranks (getranks (xml-getnodes contestants
                                        (concatenate 'string
                                                     ranktype
                                                     "Rank")))))
```

```
    (if (< 0 (length ranks))
        (concatenate 'string ranktype " Rank: " ranks)
        "")))

(export Irankings))
```