```lisp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;client-email-test.lisp
;
;Created By: Wesley R. Howell
;
;This file will contain the Theorems and test for the client-email logic
;definitions in the client-email.lisp file
;
;--Change Log-------------------------------------------------------------
;
; 3/25/2013 - Finished Theorems and Proofs. All Proofs Admit
; 3/14/2013 - File added to SVN and initial theorems proven
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(include-book "client-email")

;(contactStructure-not-nil)
;Theorem to prove that a returned contatct structure cannot be nil
;if a correct string is passed into the function
(defthm contactStructure-not-nil
  (implies (stringp str)
           (equal (listp (getContactStructure str))
                         t)))

;(contactStructure-contents-are-strings)
;Theorem to prove that the contents of the atoms in a contact structure are
;strings
(defthm contactStrucutre-contents-are-strings
  (implies (stringp str)
           (AND(equal (stringp (car (getContactStructure str))) t)
               (equal (stringp (car (cdr (getContactStructure str)))) t))
               ))

;(contactStructure-domain-is-a-string)
;Theorem to prove that the contact structure's name is a string, the above
;Theorem contactStructure-contents-are-strings builds off this proof.
(defthm contactStructure-domain-is-a-string
  (implies (stringp str)
           (equal (stringp (car (cdr (getContactStructure str))))
                  t)))

;(emailStructure-is-a-list)
;Theorem to prove that the emailStructure is returned from the function
;getEmailStructure
(defthm emailStrucutre-is-a-list
  (implies (listp xml)
           (equal (listp (getEmailStructure xml)) t)))

;(emailStrucutre-ToField-a-list
;Theorem to prove the email's to structure is a list based on the
;data structure based on the project documentation.
(defthm emailStructure-ToField-a-list
  (implies (listp xml)
           (equal (listp (car (getEmailStructure xml)))
                  t)))


;(toStringTakesListAndConvertsToString
;Theorm to test conservation of string properties for this
;function, similar to the chrs->string in the list utils
(defthm toStringTakesListAndConvertsToString
  (implies (listp toString)
           (stringp (splitToString toString)
                    )
           )
  )
```

```
;(splitToStructisList)
;Theorem to prove that the item returned is a list
;of strings that is not empty
(defthm splitToStructureList
  (implies (listp toList)
           (AND
            (listp (splitToStruct toList))
            (stringp (car (splitToStruct toList))))
   )
)


;(splitToFieldReturnsListofTos)
;Theorem to show that the splitToField function takes
;The inputed email addresses and parses the string into
;a list of valid email addresses (as strings).
;The returned type will be the same as the above theorem
;Note: The above theorm is encapsulated here as well.
(defthm splitToFieldReturensListofTos
  (implies (stringp toString)
           (AND (listp
                 (splitToField toString))
                (stringp
                 (car (splitToField toString))))
           ))


;(genEmailProducesMessage)
;Theorm to test that the email structure
;returned from generateEmailFromStrings is the correct
;email structure ((list) str str str)
(defthm genEmailProducesMessage
  (implies (AND (stringp msg)
                (AND (stringp sub)
                (AND (stringp to)
                (stringp from))))
                (AND (listp
                 (generateEmailFromStrings to from sub msg))
                (listp
                 (car (generateEmailFromStrings to from sub msg))))
           ))


;(multRecipProducesXMLList)
;Theorm to prove that the output is correct on multRecip
;based on the toList
;The format will be (XML_1, XML_2, ... , XML_n) where XML
;is the prepared string format from the getEmailXML function.
(defthm multRecipProducesXMLList
  (implies (AND (listp toList)
                (AND (stringp from)
                     (AND (stringp sub)
                          (stringp msg))))
           (AND (listp (multRecip toList from sub msg))
                (stringp (car
                          (multRecip toList from sub msg))))))


;(emailEntryPoint)
;Tests the email function since it is the entry point for the client
;side test. This theorem will need to prove that the subsequent calls
;to the email functions returns the correctly formatted list in order.
;for the IO module to output correctly. Hence, we need to prove the same steps
;as in multRecip with the extra constraint that "to" is a string.
(defthm emailEntryPoint
  (implies (AND (stringp to)
                (AND (stringp from)
                     (AND (stringp sub)
                          (stringp msg))))
           (AND (listp (email toList from sub msg))
                (stringp (car
                          (email to from sub msg))))))


;(emailTextReturnsListofString)
;Test that the email text returns a list built strings for either,
;console display or output.
(defthm emailTextReturnsListofString
  (implies (listp tokxml)
           (AND (listp (getEmailText tokxml) )
                (stringp (car (getEmailText tokxml))))))


;(emailHTMLReturnsListofStrings)
;Test that the email html output is a correct list for the IO
;funciton. This will be the pre
(defthm emailHTMLReturnsListofString
  (implies (listp tokxml)
           (AND (listp (getEmailHTML tokxml) )
                (stringp (car (getEmailHTML tokxml))))))
```