```
function compare_inputs_value, value0, value1, tagname

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Compares two values when given a tagname
;; For floating point values, an acceptable tolerance is specified.
;; Otherwise, the values have to be exact
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

same = array_equal(value0, value1) ;; first do quick check
if ~(same) then begin  ;; Now check to see if things are close
  s = (n_elements(value0) EQ n_elements(value1))
  if (s) then begin
    case strlowcase(tagname) of
      'phi': tol = 3.0*!dtor  ;; 5 degree tolerance
      'taa': tol = 3.0*!dtor  ;; 5 degree tolerance
      'lifetime': tol = min([value0,value1]) * 0.05  ;; 5% tolerance
      'endtime': tol = min([value0,value1]) * 0.05  ;; 5% tolerance
      'temperature': tol = 1.*(value0 NE 0)*(value1 NE 0) ; 1 deg unless one = 0
      'outeredge': tol = min([value0,value1]) * 0.05 ;; 5% tolerance
      'longitude': tol = 1*!dtor
      'latitude': tol = 1*!dtor
      'stickcoef': tol = 0.01
      'accom_factor': tol = 0.01
      'kappa': tol = 0.1
      'diffusionlimit': tol = min([value0,value1]) * 0.01
      else: stop
    endcase
    same = (max(abs(value0-value1)) LT tol)
  endif else same = 0  ;; different number of elements
endif

return, same

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

function compare_inputs, input0temp, input1temp, verbose=verbose

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Make a comparison between two input files
;;
;; Version 3.0: 21 Dec 2010
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if (verbose EQ !null) then verbose = 0
```

```
 52
 53  input0 = (isa(input0temp, 'string')) ? inputs_restore(input0temp) : input0temp
 54  input1 = (isa(input1temp, 'string')) ? inputs_restore(input1temp) : input1temp
 55
 56  ;; Check top level structure
 57  t0 = tag_names(input0) & t1 = tag_names(input1)
 58  q0 = sort(t0) & q1 = sort(t1)
 59
 60  same = 1
 61  if (array_equal(t0[q0], t1[q1])) then begin
 62    ;; Compare next level down
 63    i = 0
 64    while ((i LT n_elements(t0)) and (same)) do begin
 65      if (~strcmp(t0[q0[i]], t1[q1[i]])) then stop
 66      struct0 = input0.(q0[i]) & struct1 = input1.(q1[i])
 67      s0 = tag_names(struct0) & s1 = tag_names(struct1)
 68      w0 = sort(s0) & w1 = sort(s1)
 69
 70      if (array_equal(s0[w0], s1[w1])) then begin
 71        ;; Compare values of each tag
 72        j = 0
 73        while ((j LT n_elements(s0)) and (same)) do begin
 74          if (~strcmp(s0[w0[j]], s1[w1[j]])) then stop
 75
 76          ;; make sure types are the same, lengths are the same, values are the same
 77          value0 = struct0.(w0[j]) & value1 = struct1.(w1[j])
 78          type0 = size(value0, /type) & type1 = size(value1, /type)
 79          case (1) of
 80            (type0 NE type1): same = 0
 81            (type0 EQ 10): $ ;; value0 and value1 are pointers
 82              same = compare_inputs_value(*value0, *value1, s0[w0[j]])
 83            (type0 EQ 4) or (type0 EQ 5): $ ;; value0 and value1 are floats
 84              same = compare_inputs_value(value0, value1, s0[w0[j]])
 85            else: same = array_equal(value0, value1) ;; byte or integer
 86          endcase
 87          if (~same and verbose) then print, 'failed at = ' + s0[w0[j]], xxx
 88          j++
 89        endwhile
 90        i++
 91      endif else begin
 92        same = 0
 93        if (verbose) then print, 'failed at ' + t0[q0[i]], xxx
 94      endelse
 95    endwhile
 96  endif else begin
 97    same = 0
 98    if (verbose) then begin
 99      q = strcmp(t0[q0], t1[q1])
100      w = where(q EQ 0, nw)
101      print, 'Failed at Top Level', xxx
102      for i=0,nw-1 do $
```

```
103         print, '  input0.' + t0[q0[w[i]]] + '; input1.' + t1[q1[w[i]]], xxx
104       endif
105     endelse
106
107     return, same
108
109     end
```