```
; Exercise 2.3
; Solve Ex. 2.2, this time by checking for numberhood as well. Your program should
; return NIL if any (or both) of the numbers is not a number. Do NOT use AND.

(defun func1 (x y)

  (if (or (not (numberp x))    (not (numberp y))  )
      nil
      (+ x y)
      )
  )



;  Exercise 2.4

; Define a procedure that takes three numbers and returns T if all the three are integers
; and returns NIL otherwise. Do NOT use AND.

(defun func2 (x y z)

  (or  (not (integerp x))  (not (integerp y))   (not (integerp z)) )
  nil
  t
  )


; Exercise 2.5

(defun howcompute (x y z)

  (cond

   ((=  (- x y) z) (print "subtracted"))
   ((=  (- y z) z) (print "subtracted"))
   ((=  (+ x y) z) (print "add"       ))
   ((=  (* x y) z) (print "multiplied"))
    (t             (print "dont know" ))
    )
  )


; Exercise 2.6

(defun func3 (x y)

  (if (and (numberp x)  (numberp y) )
      (if (>= x y)
          x
          y
          )
      nil
      )
  )

; Exercise 2.7

(defun func4 (x y z)

  (func3 (func3 x y) z)
```

```
 61    )
 62
 63
 64  ; Exercise 2.8
 65
 66  (defun func5  (x y z)
 67
 68    (if (and (<= x y) (<= x z) )
 69        (if (<= y z)
 70            y
 71            z
 72            )
 73        (if (and (>= x y) (>= x z) )
 74            (if (>= y z)
 75                y
 76                z
 77                )
 78            x
 79            )
 80        )
 81    )
 82
 83
 84
 85  ; Exercise 2.11
 86
 87  (defun halver (x)
 88
 89    (if (< x 1)
 90        x
 91        (let  ( (h (/ x 2)) )
 92               ( halver h)
 93        )
 94    )
 95  )
 96
 97
 98  ; Exercise 2.12
 99
100  ; Rewrite (AND X Y Z W) by using cond COND.11
101
102  (defun func12 (x y z w)
103
104    (cond
105    (X
106     (cond
107       (Y
108        (cond
109          (Z
110           (cond
111             (W t)
112             (t nil)))
113         (t nil)))
114     (t nil)))
115    (t nil))
116    )
117
118
119  ; Exercise 2.13
120
```

```
; Write COND statements equivalent to: (NOT U) and (OR X Y Z)

(defun myfunc13 (x)

  (cond (x   (not x) )
        (t      t    )
        )
  )

; (OR X Y Z)

(defun myfunc13-2 (x y z)

  (cond
    (X t)
    (Y t)
    (Z t))
  )
```