; List of lists of lists of .... : Recursive Processing Part I
; =================================================

(define lolos (list (list "cat"
  "dog"
  "snail")
  (list "rhinoceros"
  "hippo")
  (list "panda"
  "monkey")
  (list "dolphin"
  "porpoise")))

(check-expect (list? lolos) #true)
(check-expect (string? lolos) #false)
(check-expect (length lolos) 4)
(check-expect (first lolos) (list "cat"
  "dog"
  "snail"))

(check-expect (list? (first lolos)) #true)
(check-expect (string? (first lolos)) #false)
(check-expect (length (first lolos)) 3)
(check-expect (string? (first (first lolos))) #true)

(require picturing-programs)
; string -> image
; Dark blue image of the text in 'a-string' using a font of size 20.
(define (string->image a-string)
  (text a-string 20 "darkblue"))

; 'check-expect's that rely on the 'text' function might fail on different computers, because the
; font might be different. But this gives you the idea:

(check-expect (string->image "cat") cat)
(check-expect (string->image (first (first lolos))) cat)
(check-expect (string->image (second (first lolos))) dog)
(check-expect (string->image (third (first lolos))) snail)
(check-expect (string->image (list-ref (first lolos) 0)) cat)
(check-expect (string->image (list-ref (first lolos) 1)) dog)
(check-expect (string->image (list-ref (first lolos) 2)) snail)

(check-expect (map string->image (first lolos)) (list cat dog snail))

; Exercise. Write an expression using 'lolos' that produces the string "catdogsni".

(check-expect (apply beside (list cat dog snail))
  catdogsni)

(check-expect (apply beside (map string->image (first lolos)))
  catdogsni)
; encircle : image -> image
; Version of 'an-image' with a dark orange ellipse around it.
(define (encircle an-image)
  (overlay an-image
       (ellipse (round (* 1.25 (image-width an-image)))
                (round (* 1.75 (image-height an-image)))
                "outline"
                "darkorange")))

; By the way, here's what a quick web search turned up when searching for: blue orange film

(check-expect (encircle (apply beside (map encircle (map string->image (first lolos)))))


(check-expect (list-of-strings->image (list "cat"
                                       "dog"
                                       "snail")))

(check-expect (list-of-strings->image (first lolos))
  (encircle (apply beside (map encircle (map string->image (first lolos)))))

(check-expect (list-of-strings->image (list "cat"
                                       "dog"
                                       "snail"))
  (encircle (apply beside (map encircle (map string->image (first lolos))))))))

; list-of-strings->image : list-of-strings -> image
(define (list-of-strings->image los)
  (encircle
   (apply beside
             (map encircle
                  (map string->image
                      los)))))

; Unlike in the introduction to lists of lists, I made sure to put at least two strings in every
; list of strings, because 'beside' in Intermediate Student complains when it's used with zero
; or one arguments/parameters/inputs.
(check-expect (map encircle (map string->image (list))) (list)

(check-expect (map encircle (map string->image (list "dolphin"))) (list)

; If you want to use a more permissive 'beside', here's one you can use with any list of images
; [but only with a *list* of images, not with individual images].

; Beside : list-of-images -> image
; (define (Beside list-of-images)
  (apply beside (square 0 "solid" "transparent") (square 0 "solid" "transparent") list-of-images))
(check-expect (Beside (list) )) ; There's an invisible image I pasted in there by using 'Beside'.

(dolphin  dolphin)

(check-expect (Beside (list cat  dog  snail))

       catdogs

; Exercise: write 'list-of-strings->image' using 'Beside' [you need to change 'beside' to 'Beside'
; but also make one other change].

; Exercise: re-implement 'Beside' using 'list*', 'append', or 'cons' to include the two invisible
; squares in the 'list-of-images' before applying 'beside'.

(check-expect (map list-of-strings->image lolos)

   (list

       cat  dog  snail

       rhinoceros  hippo

       panda  monkey

       dolphin  porpoise

   ))

(check-expect (scale 1/2

   (encircle

     (apply beside

       (map list-of-strings->image lolos))))

   (list

       cat  dog  snail  rhinoceros  hippo  panda  monkey  dolphin  porpoise

   )
