```
#|

Today we learn the How to Design Functions (HtDF) recipe.

 - it's the core of the course
 - you will use it 100s of times
 - each time you'll understand it better

Note that design methods (like the HtDF recipe):

 - make easy problems more cumbersome
 - make hard problems easier
 - make really hard problems possible

So that means you have to be patient with HtDF.

 - at first it will seem like it gets in the way
 - later it will be essential to solving the problems


|#
```

```
(@htdf topple)
(@signature Image -> Image)
;; produce image rotated by 90 degrees
(check-expect (topple (rectangle 10 20 "solid" "red"))
              (rectangle 20 10 "solid" "red"))
(check-expect (topple (triangle 20 "solid" "red"))
              (rotate 90 (triangle 20 "solid" "red")))

;(define (topple img) empty-image) ;stub

(@template-origin Image)

(@template
 (define (topple img)
   (... img)))

(define (topple img)
  (rotate 90 img))
```

```
(@htdf checkbox-line)
(@signature String -> Image)
;; produce image of box next to text
(check-expect (checkbox-line "")
              (beside (square 20 "outline" "black")
                      (text "" 20 "black")))
(check-expect (checkbox-line "apples")
              (beside (square 20 "outline" "black")
                      (text "apples" 20 "black")))
(check-expect (checkbox-line "oranges")
              (beside (square 20 "outline" "black")
                      (text "oranges" 20 "black")))

;(define (checkbox-line s) empty-image) ;stub

(@template-origin String)

(@template
 (define (checkbox-line s)
   (... s)))

(define (checkbox-line s)
  (beside (square 20 "outline" "black")
          (text s 20 "black")))
```

```
(@htdf tall?)
(@signature Image -> Boolean)
;; produce true if image is tall (height > width)
(check-expect (tall? (rectangle 10 20 "outline" "black")) true)
(check-expect (tall? (rectangle 10 10 "outline" "black")) false)
(check-expect (tall? (rectangle 10 11 "outline" "black")) true)
(check-expect (tall? (rectangle 30 20 "outline" "black")) false)

;(define (tall? i) false) ;stub

(@template-origin Image)

(@template
 (define (tall? i)
   (... i)))

(define (tall? i)
  (> (image-height i) (image-width i)))
```

```
(@htdf image>)
(@signature Image Image -> Boolean)
;; produce true if i1 is larger than i2 (comparing areas with >)
(check-expect (image> empty-image empty-image) false)
(check-expect (image> (rectangle 10 21 "outline" "black")
                      (rectangle 10 20 "outline" "black"))
              true)
(check-expect (image> (rectangle 10 20 "outline" "black")
                      (rectangle 10 20 "outline" "black"))
              false)
(check-expect (image> (rectangle 10 19 "outline" "black")
                      (rectangle 10 20 "outline" "black"))
              false)

;(define (image> i1 i2) false)

(@template-origin Image)

(@template
 (define (image> i1 i2)
   (... i1 i2)))

(define (image> i1 i2)
  (> (* (image-width i1) (image-height i1))
     (* (image-width i2) (image-height i2))))
```