Question #1. Write out the following steps. ★

Start by underlining which part(s) of the stepped expression will change, rather than copying the expression just to under-line part of it. Omit "★ Steps ★", "•", and "○".

(step (map function? (list "ant" #true list flip △)))

(step (map text? (list text-length "ant" scale △ 25)))

(step (map list? (list "ant" list #false - 25)))

(step (map unary? (list list? list length inc)))

(step (map binary? (list list? list length inc)))

(step (map triangle (list 20 25 10)))

(step (combine filled-oval (list 25 10)))

(step (map list (list "ant" #true list flip △)))

(step (combine list (list text-length "ant" scale △ 25)))
(step (map thinner (list ◆ ● ▼)))

(step (combine align-lefts (map mirror (list ◆ ● ▼))))

(step (map + (list 20 25 10)))

(step (combine + (list 20 25 10)))

(step (map * (list 20 25 10)))

(step (combine * (list 20 25 10)))

(step (map - (list 20 25 10)))

(step (combine - (list 25 10)))

(step (combine / (list 20 10)))

(step (map dec (list 20 25 10)))
: ★ Question #2. Assume the following definitions have been entered/run ...
(define s (filled-square 10))
(define c (circle 16))
(define r (combine rectangle (list 30 10)))
(define l (list s c r))

; ... then write out the following steps (in the same manner as in Question #1) ...
(step (list s c r))

(step (map filled-circle (map height (map shorter l))))

(step (combine above (map text->image (map text-join (list "ant" "bear" "ox")))))

(step (above (text->image (combine text-join (list "ant" "bear" "ox")))))

(step (combine beside (map taller (map square (map inc (map text-length (list "ant" "bear" "ox")))))))