; Question 1 [8 Marks]

; Assume the following predicate P has been defined.
;
; P : boolean boolean boolean → boolean

(define (P a b c)
  (or (not c) (and b a)))

; Part (A) [2 Marks]
;
; ★ In the definition of P above: draw an arrow from each parameter/place-holder in the header
; of the definition to where it is used in the body of the definition.

; Part (B) [6 Marks]
;
; ★ Evaluate the following expressions, showing the Intermediate Step Expressions and
; Final Result Value.

(P #false #true #true)

(P #true #true #true)

(P #false #false #false)
Question 2 [10 Marks]

Reminder: (check-expect (flip-vertical 😸) 🐾)

★ Evaluate the following expressions, showing the Intermediate Step Expressions and Final Result Value.

(scale (- (string-length "hi!") 1) (triangle 10 "outline" "black"))

(map flip-vertical (list (above □ ⬇️) □ ⬇️))

(map image? (rest (list (+ 1 2) ⊗ "image" (triangle 10 "outline" "black"))))

(apply * (range 1 4 1))
; Question 3 [8 Marks]
;
; Complete each of the two functions 'ask-answer' and 'tall?' by writing:
; ★ another correct check-expect expression
; ★ the body of the function

; Part (A) [4 Marks]

(check-expect (ask-answer "dog" "cat")
  "dog? dog? cat.")

; ★ Write another correct check-expect expression for 'ask-answer' here:

; ★ ask-answer : string string → string
;
(define (ask-answer string-1 string-2)

)

; Part (B) [4 Marks]

(check-expect (tall? 🏝)
  (> (image-height 🏝) (image-width 🏝)))

; ★ Write another correct check-expect expression for 'tall?' here:

; ★ tall?: image → boolean
;
; Is the image taller than it is wide?
;
(define (tall? an-image)

)
Question 4 [8 Marks]

Reminder: (check-expect (rotate 20) ()

Design and implement a function 'eary' by following the steps below.
Do NOT draw any images by hand: use the variable 'a-cat' instead.

(define a-cat

; ★ Write an expression, using the variable 'a-cat', that produces

; ★ Write an expression, using the variable 'a-cat' that produces

; Here is a Documentation/Test check-expect for 'eary':

(check-expect (eary (a-cat) ) )

; ★ Complete this check-expect expression, using the variable 'a-cat':

(check-expect (eary (a-cat) ) )

; ★ Fill in the header and body to define the function 'eary'.

; eary : image → image

(define (define}