**Structures and Interpretation of Computer Program**

**Exercise Chapter 2.1 Name:** Wan Huzaifah bin Wan Azhar

**Exercise 2.2.4 Example: A picture Language**



(define (up-split painter n)

(if (= n 0)

painter

(let ((smaller (up-split painter (- n 1))))

(below painter (beside smaller smaller)))))

(define (split op1 op2)

(lambda (painter n)

(if (= n 0)

painter

(let ((smaller ((split op1 op2) painter (- n 1))))

(op1 painter (op2 smaller smaller))))))



(define (make-vect x y)

(cons x y))

(define (xcor-vect vect)

(car vect))

(define (ycor-vect vect)

(cdr vect))

(define (add-vect vect1 vect2)

(make-vect (+ (xcor-vect vect1) (xcor-vect vect2))

(+ (ycor-vect vect1) (ycor-vect vect2))))

(define (sub-vect vect1 vect2)

(make-vect (- (xcor-vect vect1) (xcor-vect vect2))

(- (ycor-vect vect1) (ycor-vect vect2))))

(define (mult-vect vect s)

(make-vect (\* s (xcor-vect vect))

(\* s (ycor-vect vect))))



(define (make-frame origin edge1 edge2)

(list origin edge1 edge2))

(define (origin-frame frame)

(car frame))

(define (edge1-frame frame)

(cadr frame))

(define (edge2-frame frame)

(caddr frame))

(define (make-frame-two origin edge1 edge2)

(cons origin (cons edge1 edge2)))

(define (origin-frame-two frame)

(car frame))

(define (edge1-frame-two frame)

(cadr frame))

(define (edge2-frame-two frame)

(cddr frame))



;Start-point and end-point is vector

(define (make-segment start-point end-point)

(cons start-point end-point))

(define (start-segment segment)

(car segment))

(define (end-segment segment)

(cdr segment))



(define (painter->outline frame)

(let ((bl (make-vect 0 0))

(br (make-vect 1 0))

(tl (make-vect 0 1))

(tr (make-vect 1 1)))

(segments->painter (list (make-segment bl br)

(make-segment br tr)

(make-segment bl tl)

(make-segment tr tl)))

frame))

(define (painter->XCorner frame)

(let ((bl (make-vect 0 0))

(br (make-vect 1 0))

(tl (make-vect 0 1))

(tr (make-vect 1 1)))

(segments->painter (list (make-segment bl tr)

(make-segment br tl)))

frame))

(define (painter->diamond frame)

(let ((bm (make-vect 0.5 0.0))

(tm (make-vect 1.0 0.5))

(lm (make-vect 0.0 0.5))

(rm (make-vect 1.0 0.5)))

(segments->painter (list (make-segment bm rm)

(make-segment rm tm)

(make-segment tm lm)

(make-segment lm bm)))

frame))



(define (flip-horiz painter)

(transform-painter painter

(make-vect 1.0 0.0)

(make-vect 0.0 0.0)

(make-vect 1.0 1.0)))

(define (rotate180 painter)

(transform-painter painter

(make-vect 1.0 1.0)

(make-vect 1.0 0.0)

(make-vect 0.0 1.0)))

(define (rotate270 painter)

(transform-painter painter

(make-vect 0.0 1.0)

(make-vect 0.0 0.0)

(make-vect 1.0 1.0)))



(define (below painter1 painter2)

(let ((split-point (make-vect 0.0 0.5)))

(let ((paint-bottom

(transform-painter painter1

(make-vect 0.0 0.0)

(make-vect 0.0 1.0)

split-point)))

((paint-top

(transform-painter painter2

split-point

(make-vect 0.0 1.0)

(make-vect 1.0 0.0))))

(lamba (frame)

(paint-bottom frame)

(paint-top frame)))))

(define (below-alt painter1 painter2)

(rotate90 (beside (rotate270 painter1) (rotate270 painter2))))



(define (corner-split painter n)

(below (beside painter (right-split painter n)))

(beside (up-split painter n) (corner-split painter n)))

(define (square-limit painter n)

(let ((combine4 (square-of-four flip-vert rotate180

identity flip-horiz)))

(combine4 (corner-split painter n))))