

```

;;Ross Lagoy & Hailey Presscot, Section B

;; contract: dis: posn posn -> number 270 30
;; purpose: to compute half of the distance between two posns
;; example: (mid 4 8) should be 6
(define (dis a b)
  (sqrt (+ (expt (- (posn-x a) (posn-x b)) 2) (expt (- (posn-y a) (posn-y b)) 2))))

;;contract: draw-line: posn posn -> image
;;purpose: draw a solid black line using two points
;;examples: (draw-solid-line (make-posn 0 0) (make-posn 100 0) should produce a solid line with
two identified endpoints
(define (draw-line aposn bposn)
  (and (draw-solid-line aposn bposn 'green)
       (draw-solid-line bposn aposn 'green)))

;; contract: circle-pt: number number posn -> posn
;; purpose: given the angle, radius and center of the circle, give the point on the circle at
that angle
;; example: (circle-pt 120 100 (make-posn 100 100)) should return (make-posn 50 186.6)
(define (circle-pt n radius center)
  (make-posn (+ (posn-x center) (* radius (cos (* 2 pi (/ n 360)))))
             (+ (posn-y center) (* radius (sin (* 2 pi (/ n 360)))))))

;; contract: too-small?: posn posn -> boolean
;; purpose: to determine if the distance between the points is too small to continue recursion
;; example: (too-small? (make-posn 0 0) (make-posn 100 0) (make-posn 100 100)) should return
false
(define (too-small? a b)
  (and (< (abs (- (posn-x a) (posn-x b))) 2) (< (abs (- (posn-y a) (posn-y b))) 2)))

;;contract: cons-point: posn posn number -> posn
;;purpose: to construct a point at a given angle and distance away from aposn
;;example: (cons-point (aposn bposn 270)) should produce "a point"
(define (cons-point aposn bposn angle right?)
  (cond [(boolean=? right? true) (circle-pt (+ angle 30) (* (dis aposn bposn) .67) bposn)]
        [else (circle-pt (- angle 30) (* (dis aposn bposn) .67) bposn)]))

;;construct: ftree: posn posn number-> image
;;purpose: given two posns and the desired angle, draw a fractal tree
;;examples: (ftree ((make-posn 200 150) (make-posn 200 300) 270) should produce a fractal tree
with the starting posns and angle (200,150), (200,300), and 270
(define (ftree aposn bposn angle)
  (cond [(too-small? aposn bposn) true]
        [else (and (draw-line aposn bposn)
                    (sleep-for-a-while .000001)
                    (ftree bposn (cons-point aposn bposn angle false) (- angle 30))
                    (ftree bposn (cons-point aposn bposn angle true) (+ angle 30))))]))

;ftree;
(start 1000 1000)
(ftree (circle-pt 270 340 (make-posn 500 1100))
      (circle-pt 270 340 (make-posn 500 900))
      270)

```

