Code to calculate the circle that passes through three given points.

Name: distance

Computes the distance between two points.

inputs:

-point0x: a float representing the x-coordinate of the first point

-point0y: a float representing the y-coordinate of the first point

-point1x: a float representing the x-coordinate of the second point

-point1y: a float representing the y-coordinate of the second point

returns: a float that is the distance between the two points

dist ← ((point0x - point1x)^2 + (point0y - point1y)^2)^.5

return dist

Name: midpoint

Computes the midpoint between two points.

inputs:

-point0x: a float representing the x-coordinate of the first point

-point0y: a float representing the y-coordinate of the first point

-point1x: a float representing the x-coordinate of the second point

-point1y: a float representing the y-coordinate of the second point

returns: two floats that are the x- and y-coordinates of the midpoint

midpointx ← point0x + (point1x - point0x)/2

midpointy ← point0y + (point1y - point0y)/2

return midpointx, midpointy

Name: slope

Computes the slope of the line that connects two given points.

The x-values of the two points, point0x and poin1x, must be different.

inputs:

-point0x: a float representing the x-coordinate of the first point.

-point0y: a float representing the y-coordinate of the first point

-point1x: a float representing the x-coordinate of the second point.

-point1y: a float representing the y-coordinate of the second point

returns: a float that is the slope between the points

lineslope ← (point1y - point0y) / (point1x - point0x)

return lineslope

Name: perp

Computes the slope of a line perpendicular to a given slope.

input:

-lineslope: a float representing the slope of a line.

Must be non-zero

returns: a float that is the perpendicular slope

perpslope ← -1 / lineslope

return perpslope

Name: intersect

Computes the intersection point of two lines.

The two slopes, slope0 and slope1, must be different.

inputs:

-slope0: a float representing the slope of the first line.

-point0x: a float representing the x-coordinate of the first point

-point0y: a float representing the y-coordinate of the first point

-slope1: a float representing the slope of the second line.

-point1x: a float representing the x-coordinate of the second point

-point1y: a float representing the y-coordinate of the second point

returns: two floats that are the x- and y-coordinates of the intersection

point

intersectx ← ((slope0\*point0x) - (slope1\*point1x) + (point1y - point0y))/(slope0-slope1)

intersecty ← slope0 \* (intersectx - point0x) + point0y

return intersectx, intersecty

Name: make\_circle

Computes the center and radius of a circle that passes through

thre given points.

The points must not be co-linear and no two points can have the

same x or y values.

inputs:

-point0x: a float representing the x-coordinate of the first point

-point0y: a float representing the y-coordinate of the first point

-point1x: a float representing the x-coordinate of the second point

-point1y: a float representing the y-coordinate of the second point

-point2x: a float representing the x-coordinate of the third point

-point2y: a float representing the y-coordinate of the third point

returns: three floats that are the x- and y-coordinates of the center

and the radius

midpoint0x, midpoint0y ← midpoint(point0x, point0y, point1x, point1y)

midpoint1x, midpoint1y ← midpoint(point0x, point0y, point2x, point2y)

slope0 ← slope(point0x, point0y, point1x, point1y)

perp0 ← perp(slope0)

slope1 ← slope(point0x, point0y, point2x, point2y)

perp1 ← perp(slope1)

centerx, centery ← intersect(perp0, midpoint0x, midpoint0y, perp1, midpoint1x, midpoint1y)

radius ← distance(centerx, centery, point0x, point0y)

return centerx, centery, radius