```
# Worksheet 8 – Points, Vectors, Segments, Lines
import math

# Q1: Point operations
class Point:
    def __init__(self,x,y):
        self.x,self.y = x,y
    def dist(self, other):
        return math.hypot(self.x-other.x, self.y-other.y)
    def midpoint(self, other):
        return Point((self.x+other.x)/2, (self.y+other.y)/2)

def line_coeffs(A,B):
    a = A.y - B.y
    b = B.x - A.x
    c = A.x*B.y - B.x*A.y
    return a,b,c

def line_eq(A,B):
    if A.x == B.x:
        print("Vertical line: x =", A.x)
    else:
        m = (B.y-A.y)/(B.x-A.x)
        c = A.y - m*A.x
        print("y =", m,"x +",c)

def reflect_point_over_line(C,A,B):
    a,b,c = line_coeffs(A,B)
    d = (a*C.x + b*C.y + c)/(a*a + b*b)
    xr = C.x - 2*a*d
    yr = C.y - 2*b*d
    return Point(xr,yr)

def demo_q1():
    x1,y1 = map(float,input("A (x1 y1): ").split())
    x2,y2 = map(float,input("B (x2 y2): ").split())
    x3,y3 = map(float,input("C (x3 y3): ").split())
    A,B,C = Point(x1,y1), Point(x2,y2), Point(x3,y3)
    print("Distance A-B:", A.dist(B))
    mid = A.midpoint(B)
    print("Midpoint:", mid.x, mid.y)
    line_eq(A,B)
    R = reflect_point_over_line(C,A,B)
    print("Reflection of C:", R.x, R.y)

# Q2: 2D vector operations
def add3(a,b,c):
    return (a[0]+b[0]+c[0], a[1]+b[1]+c[1])

def mag(v):
    return math.hypot(v[0], v[1])

def dot(u,v):
    return u[0]*v[0] + u[1]*v[1]

def angle(u,v):
    duv = dot(u,v)
    m = mag(u)*mag(v)
    if m == 0:
        return 0.0
    cos_t = max(-1,min(1,duv/m))
    return math.degrees(math.acos(cos_t))

def proj(u,v):
    dv = dot(u,v)
    mv2 = mag(v)**2
    if mv2 == 0:
        return (0.0,0.0)
    k = dv/mv2
    return (k*v[0], k*v[1])

def demo_q2():
```

```
    ax,ay = map(float,input("A (ax ay): ").split())
    bx,by = map(float,input("B (bx by): ").split())
    cx,cy = map(float,input("C (cx cy): ").split())
    A,B,C = (ax,ay),(bx,by),(cx,cy)
    R = add3(A,B,C)
    print("R = A+B+C:", R)
    print("|A|, |B|, |C|:", mag(A), mag(B), mag(C))
    print("A·B, A·C, B·C:", dot(A,B), dot(A,C), dot(B,C))
    print("angle(A,B), angle(A,C), angle(B,C):",
          angle(A,B), angle(A,C), angle(B,C))
    print("Projection of A on B:", proj(A,B))

# Q3: Segment and distance
def seg_length(S,E):
    return math.hypot(E[0]-S[0], E[1]-S[1])

def closest_point_on_segment(S,E,P):
    sx,sy = S
    ex,ey = E
    px,py = P
    vx,vy = ex-sx, ey-sy
    wx,wy = px-sx, py-sy
    vv = vx*vx + vy*vy
    if vv == 0:
        return S
    t = (wx*vx + wy*vy)/vv
    t = max(0,min(1,t))
    cx = sx + t*vx
    cy = sy + t*vy
    return (cx,cy)

def dist(P,Q):
    return math.hypot(P[0]-Q[0], P[1]-Q[1])

def demo_q3():
    x1,y1 = map(float,input("S (x1 y1): ").split())
    x2,y2 = map(float,input("E (x2 y2): ").split())
    x3,y3 = map(float,input("P (x3 y3): ").split())
    S,E,P = (x1,y1),(x2,y2),(x3,y3)
    print("Segment length SE:", seg_length(S,E))
    C = closest_point_on_segment(S,E,P)
    print("Closest point on SE to P:", C)
    print("Distance P to segment:", dist(P,C))

# Q4: Intersection of two lines
def line_intersection(a1,b1,c1,a2,b2,c2):
    D = a1*b2 - a2*b1
    if D == 0:
        return None
    x = (c1*b2 - c2*b1)/D
    y = (a1*c2 - a2*c1)/D
    return (x,y)

def demo_q4():
    a1,b1,c1 = map(float,input("L1 (a1 b1 c1): ").split())
    a2,b2,c2 = map(float,input("L2 (a2 b2 c2): ").split())
    pt = line_intersection(a1,b1,c1,a2,b2,c2)
    if pt is None:
        print("Lines are parallel or coincident.")
    else:
        print("Intersection at:", pt)

if __name__ == "__main__":
    while True:
        print("\n1: Q1 demo  2: Q2 demo  3: Q3 demo  4: Q4 demo  5: Exit")
        choice = input("Choose: ").strip()
        if choice == "1":
            demo_q1()
        elif choice == "2":
            demo_q2()
        elif choice == "3":
            demo_q3()
```

```
elif choice == "4":
    demo_q4()
elif choice == "5":
    break
else:
    print("Invalid.")
```