class Point:

def \_\_init\_\_(self, x, y, curve):

self.x = x

self.y = y

self.curve = curve

def \_\_add\_\_(self, other):

if self == other:

return self.curve.double\_point(self)

return self.curve.add\_points(self, other)

def \_\_mul\_\_(self, n):

return self.curve.multiply\_point(self, n)

def \_\_str\_\_(self):

return f"({self.x}, {self.y})"

class EllipticCurve:

def \_\_init\_\_(self, a, b, p):

self.a = a

self.b = b

self.p = p

def add\_points(self, P, Q):

if P.x == Q.x and P.y == Q.y:

return self.double\_point(P)

lam = (Q.y - P.y) \* pow(Q.x - P.x, -1, self.p) % self.p

x\_r = (lam\*\*2 - P.x - Q.x) % self.p

y\_r = (lam \* (P.x - x\_r) - P.y) % self.p

print(f"Phép cộng điểm: λ = ({Q.y} - {P.y}) \* ({Q.x} - {P.x})^(-1) mod {self.p} = {lam}")

print(f"x\_r = λ^2 - x\_P - x\_Q mod {self.p} = {x\_r}")

print(f"y\_r = λ \* (x\_P - x\_r) - y\_P mod {self.p} = {y\_r}")

return Point(x\_r, y\_r, self)

def double\_point(self, P):

lam = (3 \* P.x\*\*2 + self.a) \* pow(2 \* P.y, -1, self.p) % self.p

x\_r = (lam\*\*2 - 2 \* P.x) % self.p

y\_r = (lam \* (P.x - x\_r) - P.y) % self.p

print(f"Phép nhân đôi điểm: λ = (3 \* {P.x}^2 + {self.a}) \* (2 \* {P.y})^(-1) mod {self.p} = {lam}")

print(f"x\_r = λ^2 - 2 \* {P.x} mod {self.p} = {x\_r}")

print(f"y\_r = λ \* ({P.x} - {x\_r}) - {P.y} mod {self.p} = {y\_r}")

return Point(x\_r, y\_r, self)

def multiply\_point(self, P, n):

result = None

addend = P

print(f"Phép nhân điểm: {n} \* {P}")

while n:

if n & 1:

result = addend if result is None else result + addend

print(f"Kết quả tạm thời: {result}")

addend += addend

n >>= 1

return result

def main():

# Nhập thông tin cho elliptic curve

a = int(input("Nhập hệ số a: "))

b = int(input("Nhập hệ số b: "))

p = int(input("Nhập modulus p: "))

curve = EllipticCurve(a, b, p)

# Nhập điểm sinh G

Gx = int(input("Nhập tọa độ x của điểm sinh G: "))

Gy = int(input("Nhập tọa độ y của điểm sinh G: "))

G = Point(Gx, Gy, curve)

# Nhập khóa riêng của A và B

n\_A = int(input("Nhập khóa riêng của A: "))

n\_B = int(input("Nhập khóa riêng của B: "))

# Tính khóa công khai

P\_A = G \* n\_A

P\_B = G \* n\_B

print(f"Khóa công khai của A (P\_A): {P\_A}")

print(f"Khóa công khai của B (P\_B): {P\_B}")

# Nhập bản tin cần gửi

PMx = int(input("Nhập tọa độ x của bản tin P\_M: "))

PMy = int(input("Nhập tọa độ y của bản tin P\_M: "))

P\_M = Point(PMx, PMy, curve)

# Nhập giá trị ngẫu nhiên k

k = int(input("Nhập giá trị ngẫu nhiên k: "))

# Mã hóa bản tin

C1 = G \* k

C2 = P\_M + (P\_B \* k)

print(f"Mã hóa: C1 = {C1}, C2 = {C2}")

# Giải mã bản tin

decoded\_P\_M = C2 + Point(C1.x, -C1.y, curve) \* n\_B

print(f"Giải mã: P\_M = {decoded\_P\_M}")

if \_\_name\_\_ == "\_\_main\_\_":

main()