Lab – 7

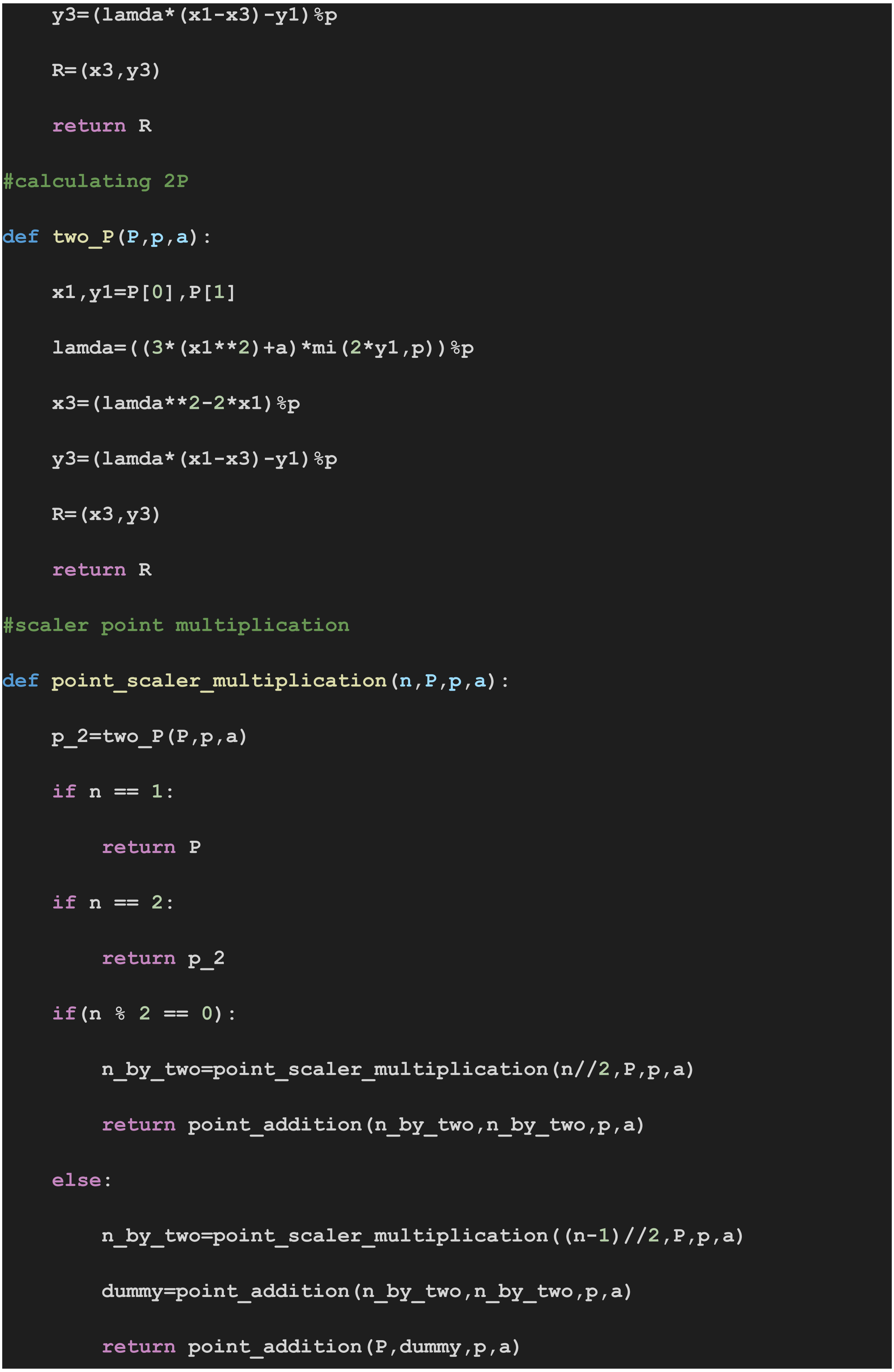
Subject : NIS

Aim :- Implement ECC point Encryption and Decryption.

1. Do Key generation
2. Encryption
3. Decryption

Program: -

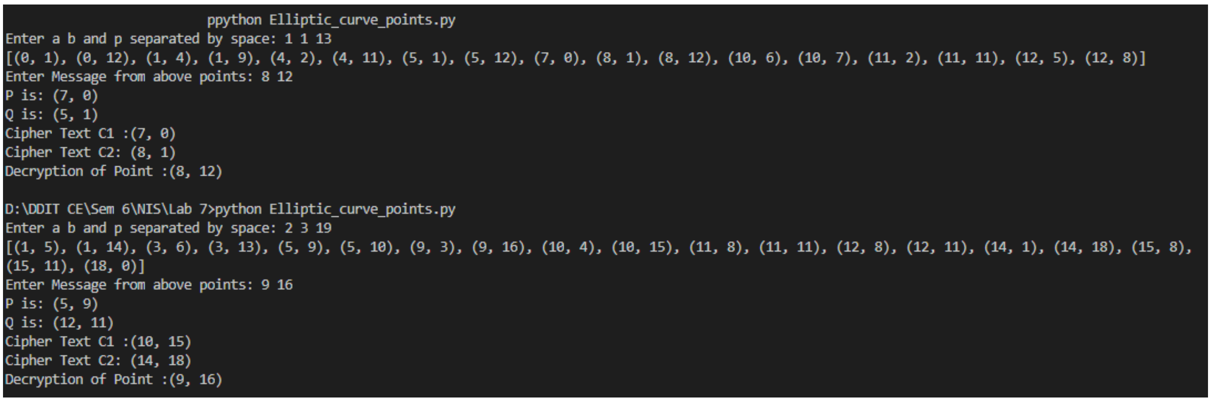
```
import math

from ExtendedEuclidian import multiplicative_inverse as mi

from random import randint

#key generation

def key_generation(P,p,a):

    e1=P

    d=randint(1,p-1)

    e2=point_scaler_multiplication(d,P,p,a)

    public=[e1,e2]

    private=d

    return public,private

#point encryption

def encrypt(M,public,p,a):

    r=randint(1,p-1)
    c1=point_scaler_multiplication(r,public[0],p,a)

c2=point_addition(M,point_scaler_multiplication(r,public[1],p,a),p,a)
    return c1,c2
#point decryption
def decrypt(c1,c2,private,p,a):
    return
point_substraction(c2,point_scaler_multiplication(private,c1,p,a),p,a)
#point substraction
def point_substraction(P,Q,p,a):
    x1,x2=P[0],Q[0]
    y1,y2=P[1],(-1)*Q[1]
    new_P=(x1,y1)
    new_Q=(x2,y2)
    return point_addition(new_P,new_Q,p,a)
#point addition
def point_addition(P,Q,p,a):
    x1,x2=P[0],Q[0]
    y1,y2=P[1],Q[1]
    if(x1 == x2 and y1 == y2):
        return two_P(P,p,a)
    if (x2-x1) < 0:
        lamda=((-1)*(y2-y1)*mi(((-1)*(x2-x1)),p))%p
    else:
        lamda=((y2-y1)*mi((x2-x1),p))%p
    x3=(lamda**2-x1-x2)%p
```



```
    y3=(lamda*(x1-x3)-y1)%p

    R=(x3,y3)

    return R

#calculating 2P

def two_P(P,p,a):

    x1,y1=P[0],P[1]

    lamda=((3*(x1**2)+a)*mi(2*y1,p))%p

    x3=(lamda**2-2*x1)%p

    y3=(lamda*(x1-x3)-y1)%p

    R=(x3,y3)

    return R

#scaler point multiplication

def point_scaler_multiplication(n,P,p,a):

    p_2=two_P(P,p,a)

    if n == 1:

        return P

    if n == 2:

        return p_2

    if(n % 2 == 0):

        n_by_two=point_scaler_multiplication(n//2,P,p,a)

        return point_addition(n_by_two,n_by_two,p,a)

    else:

        n_by_two=point_scaler_multiplication((n-1)//2,P,p,a)

        dummy=point_addition(n_by_two,n_by_two,p,a)

        return point_addition(P,dummy,p,a)
```

```
#ECC point generation
def Elliptic_curve_points(a,b,p):
    x=0
    points=[]
    while(x < p):
        w=(x**3+a*x+b)%p
        result=w**((p-1)//2) % p
        if(w == 0):
            points.append((x,0))
        if(result == 1):
            root = math.sqrt(w)
            while math.ceil(root) != root:
                w+=p
                root = math.sqrt(w)
            points.append((x,int(root%p)))
            points.append((x,int((-root)%p)))
        if(result == -1):
            print("No solution")
            break
        x+=1
    return points
#main driver program
if __name__ == "__main__":
    a,b,p=list(map(int,input("Enter a b and p separated by space: ").split()))
    list_of_points=Elliptic_curve_points(a,b,p)
    print(list_of_points)
    x,y=list(map(int,input("Enter Message from above points: ").split()))
    M=(x,y)
    l=len(list_of_points)-1
    r_p=randint(0,l)
    r_q=randint(0,l)
    P=list_of_points[r_p]
    Q=list_of_points[r_q]
    while(M ==P or M == Q):
        r_p=randint(0,l)
        r_q=randint(0,l)
        P=list_of_points[r_p]
        Q=list_of_points[r_q]
    print("P is:",P)
    print("Q is:",Q)
    public,private=key_generation(P,p,a)
    c1,c2=encrypt(M,public,p,a)
    dec_=decrypt(c1,c2,private,p,a)
    print("Cipher Text C1 :"+str(c1))
    print("Cipher Text C2: "+str(c2))
    #print(type(public),type(private))
    print("Decryption of Point :"+str(dec_))
    #print("Addition is:",point_addition((12,5),(11,11),p,a))
    #print("Multiplicatioin
is:",point_scaler_multiplication(97,(197,167),p,a))

    #print("Multiplicatioin
is:",point_scaler_multiplication(101,(94,133),p,a))
```

Output: -



```
                          ppython Elliptic_curve_points.py
Enter a b and p separated by space: 1 1 13
[(0, 1), (0, 12), (1, 4), (1, 9), (4, 2), (4, 11), (5, 1), (5, 12), (7, 0), (8, 1), (8, 12), (10, 6), (10, 7), (11, 2), (11, 11), (12, 5), (12, 8)]
Enter Message from above points: 8 12
P is: (7, 0)
Q is: (5, 1)
Cipher Text C1 :(7, 0)
Cipher Text C2: (8, 1)
Decryption of Point :(8, 12)

D:\DDIT CE\Sem 6\NIS\Lab 7>python Elliptic_curve_points.py
Enter a b and p separated by space: 2 3 19
[(1, 5), (1, 14), (3, 6), (3, 13), (5, 9), (5, 10), (9, 3), (9, 16), (10, 4), (10, 15), (11, 8), (11, 11), (12, 8), (12, 11), (14, 1), (14, 18), (15, 8),
(15, 11), (18, 0)]
Enter Message from above points: 9 16
P is: (5, 9)
Q is: (12, 11)
Cipher Text C1 :(10, 15)
Cipher Text C2: (14, 18)
Decryption of Point :(9, 16)
```