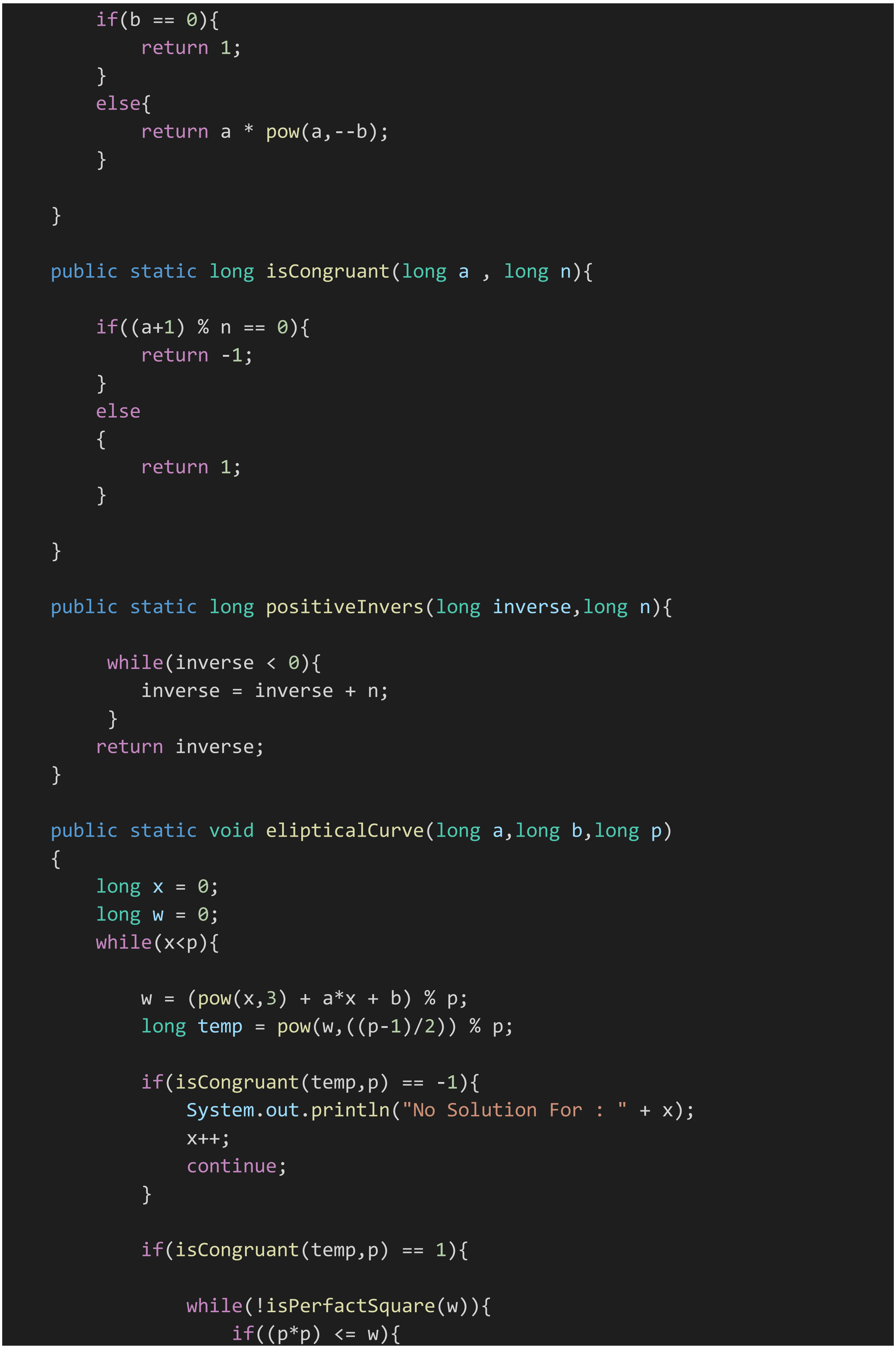
Lab – 7

Subject: NIS

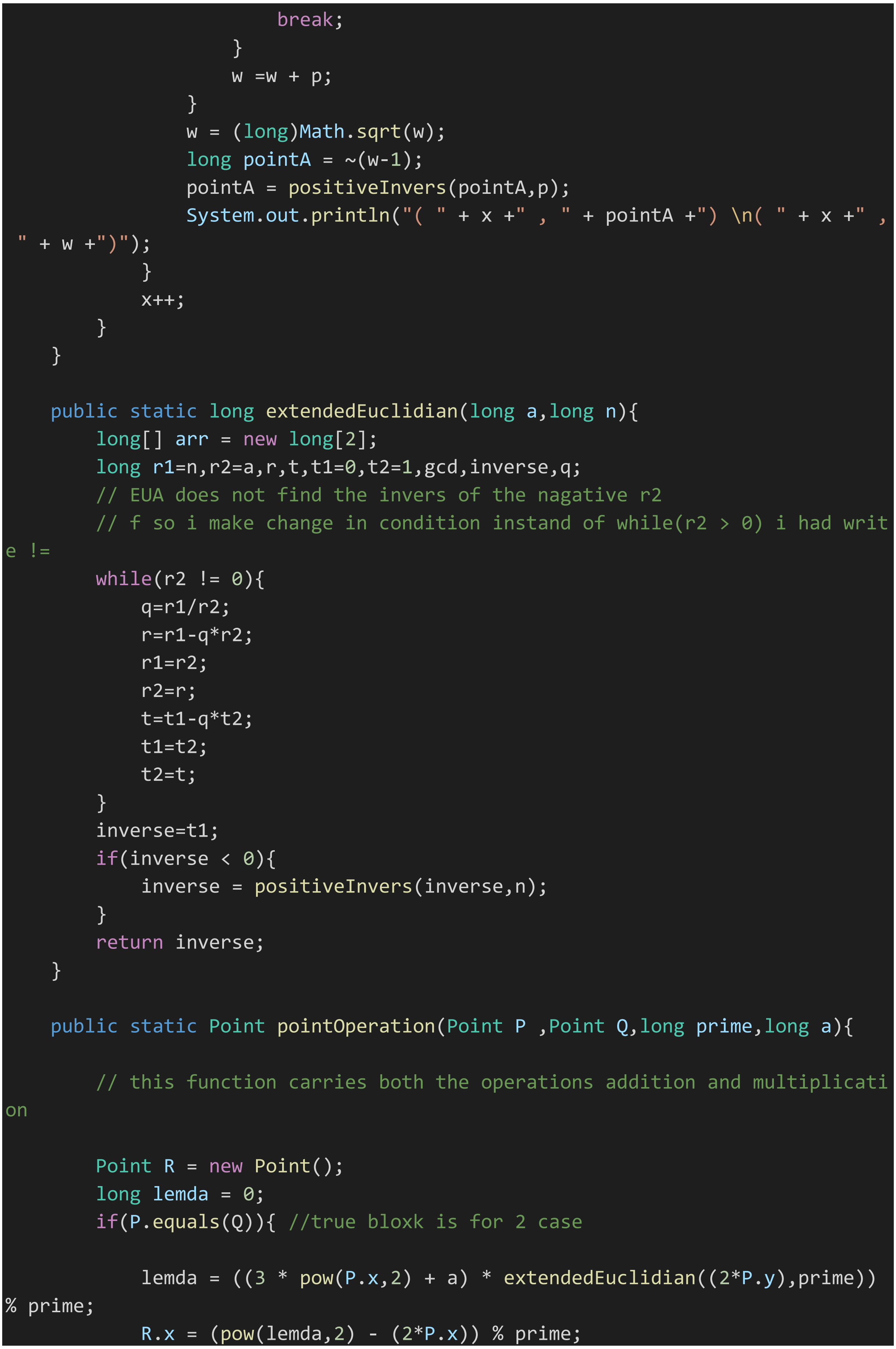
Aim: Implement Addition of two Points with respect to Elliptical Curve Cryptography (For all 3 cases). Using Addition function, find scalar multiplication i.e. multiplication of a Point by scalar (Note: O(log(n)) additions only). Using both (1) and (2), Do Key Generation for ECC Elgamal . Using Keys Encrypt the Point(On the Curve) and Decrypt the Cipher points. Show that the implementation works Correctly.

Program: -

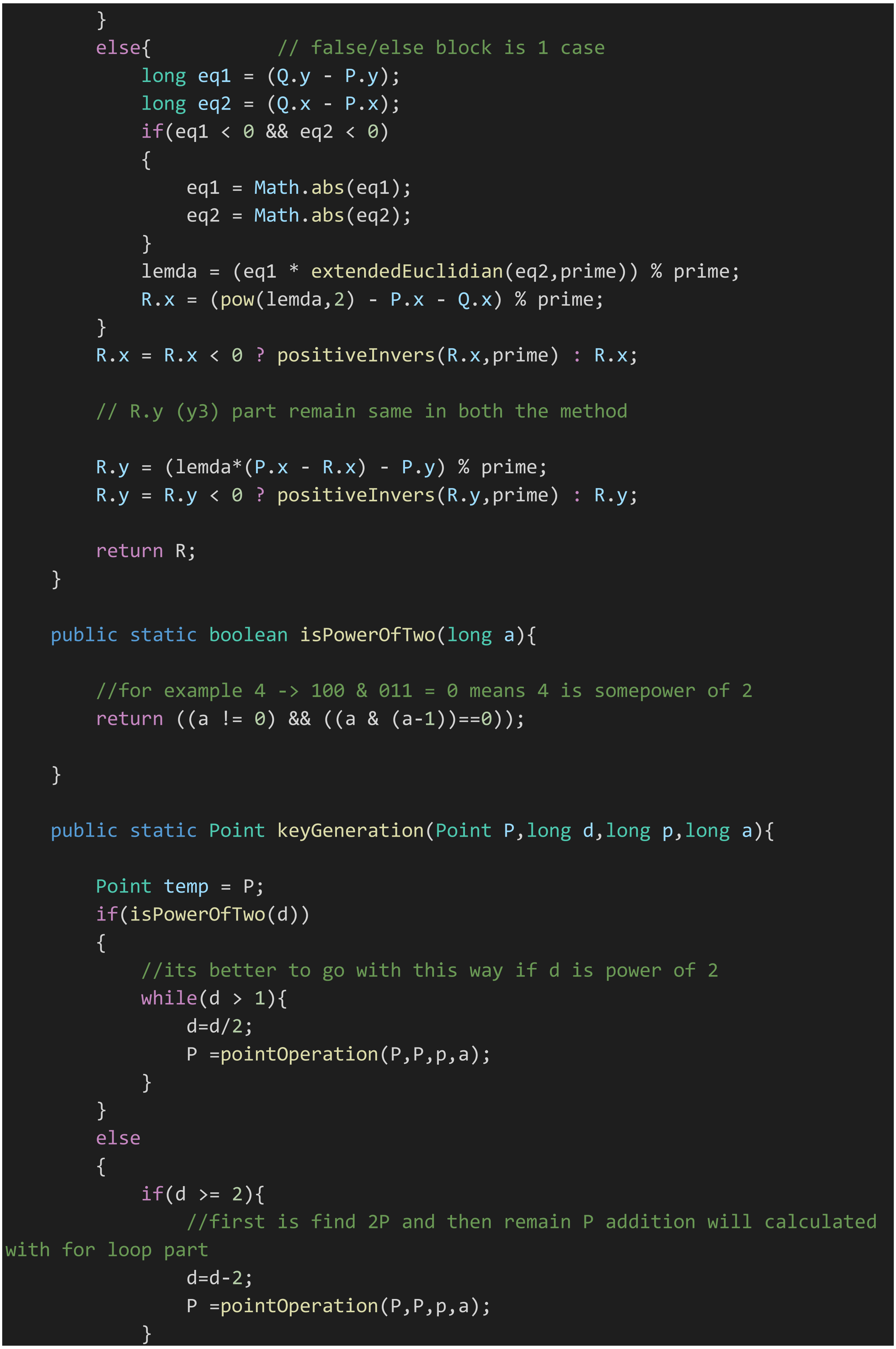
```
import java.util.*;
import java.lang.*;

class Point{

    long x = 0;
    long y = 0;

    public Point(){}
    public Point(long a,long b){
        this.x=a;
        this.y=b;
    }
    public long getX(){
        return x;
    }
    public long getY(){
        return y;
    }
    @Override
    public boolean equals(Object obj){
        //check for the object value wheter it is equal or not
        return obj != null && this.getX() == ((Point)obj).getX() && this.getY(
) == ((Point)obj).getY() ? true : false;
    }
}

public class ECC
{
    public static boolean isPerfactSquare(long n){

        double sqrt = Math.sqrt((double)n);
        return (sqrt - Math.floor(sqrt)==0);

    }

    public static long pow(long a,long b){
```



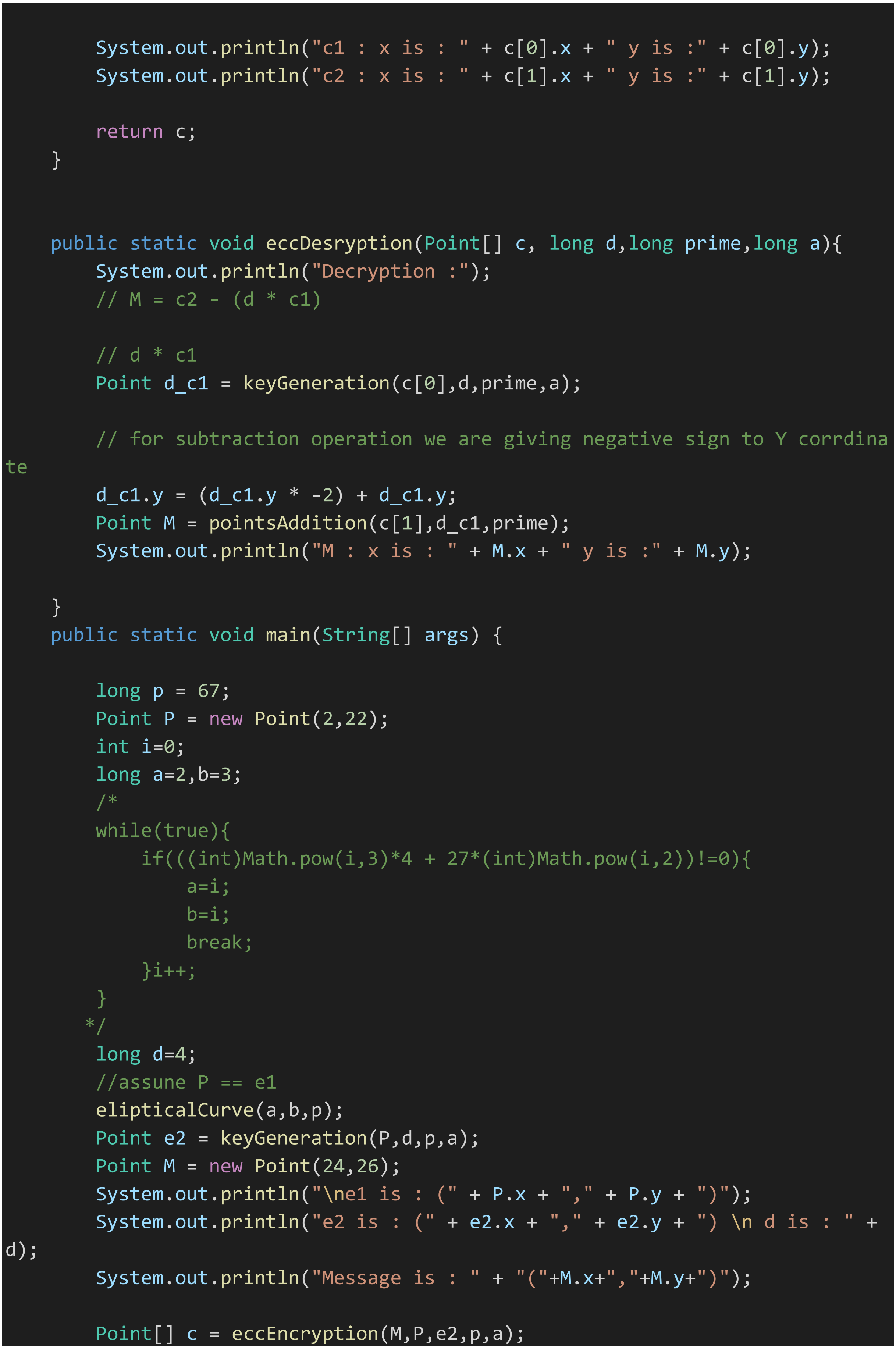
```
        if(b == 0){
            return 1;
        }
        else{
            return a * pow(a,--b);
        }

    }

    public static long isCongruant(long a , long n){

        if((a+1) % n == 0){
            return -1;
        }
        else
        {
            return 1;
        }

    }

    public static long positiveInvers(long inverse,long n){

         while(inverse < 0){
            inverse = inverse + n;
         }
        return inverse;
    }

    public static void elipticalCurve(long a,long b,long p)
    {
        long x = 0;
        long w = 0;
        while(x<p){

            w = (pow(x,3) + a*x + b) % p;
            long temp = pow(w,((p-1)/2)) % p;

            if(isCongruant(temp,p) == -1){
                System.out.println("No Solution For : " + x);
                x++;
                continue;
            }

            if(isCongruant(temp,p) == 1){

                while(!isPerfactSquare(w)){
                    if((p*p) <= w){
```



```
                        break;
                    }
                    w =w + p;
                }
                w = (long)Math.sqrt(w);
                long pointA = ~(w-1);
                pointA = positiveInvers(pointA,p);
                System.out.println("( " + x +" , " + pointA +") \n( " + x +" ,
 " + w +")");
            }
            x++;
        }
    }

    public static long extendedEuclidian(long a,long n){
        long[] arr = new long[2];
        long r1=n,r2=a,r,t,t1=0,t2=1,gcd,inverse,q;
        // EUA does not find the invers of the nagative r2
        // f so i make change in condition instand of while(r2 > 0) i had writ
e !=
        while(r2 != 0){
            q=r1/r2;
            r=r1-q*r2;
            r1=r2;
            r2=r;
            t=t1-q*t2;
            t1=t2;
            t2=t;
        }
        inverse=t1;
        if(inverse < 0){
            inverse = positiveInvers(inverse,n);
        }
        return inverse;
    }

    public static Point pointOperation(Point P ,Point Q,long prime,long a){

        // this function carries both the operations addition and multiplicati
on

        Point R = new Point();
        long lemda = 0;
        if(P.equals(Q)){ //true bloxk is for 2 case

            lemda = ((3 * pow(P.x,2) + a) * extendedEuclidian((2*P.y),prime))
% prime;
            R.x = (pow(lemda,2) - (2*P.x)) % prime;
```

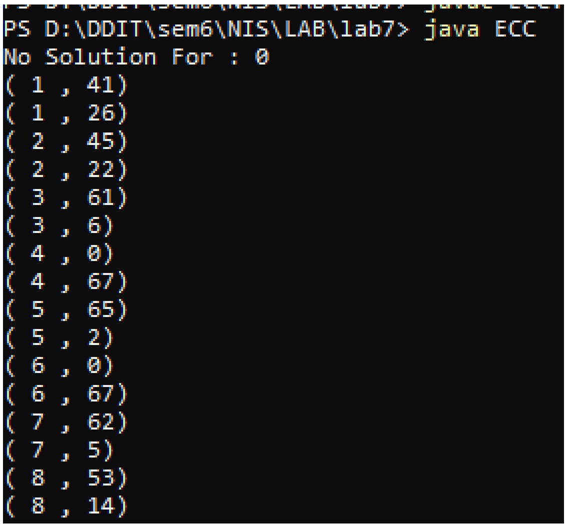


```
        }
        else{           // false/else block is 1 case
            long eq1 = (Q.y - P.y);
            long eq2 = (Q.x - P.x);
            if(eq1 < 0 && eq2 < 0)
            {
                eq1 = Math.abs(eq1);
                eq2 = Math.abs(eq2);
            }
            lemda = (eq1 * extendedEuclidian(eq2,prime)) % prime;
            R.x = (pow(lemda,2) - P.x - Q.x) % prime;
        }
        R.x = R.x < 0 ? positiveInvers(R.x,prime) : R.x;

        // R.y (y3) part remain same in both the method

        R.y = (lemda*(P.x - R.x) - P.y) % prime;
        R.y = R.y < 0 ? positiveInvers(R.y,prime) : R.y;

        return R;
    }

    public static boolean isPowerOfTwo(long a){

        //for example 4 -> 100 & 011 = 0 means 4 is somepower of 2
        return ((a != 0) && ((a & (a-1))==0));

    }

    public static Point keyGeneration(Point P,long d,long p,long a){

        Point temp = P;
        if(isPowerOfTwo(d))
        {
            //its better to go with this way if d is power of 2
            while(d > 1){
                d=d/2;
                P =pointOperation(P,P,p,a);
            }
        }
        else
        {
            if(d >= 2){
                //first is find 2P and then remain P addition will calculated
with for loop part
                d=d-2;
                P =pointOperation(P,P,p,a);
            }
            for(int i=0;i<d;i++){
                P = pointOperation(P,temp,p,a);
            }
        }
        return P;
    }

    public static Point pointsAddition(Point P , Point Q,long prime){

        //point addition

        Point R = new Point();
        long eq1 = (Q.y - P.y);
        long eq2 = (Q.x - P.x);

        if(eq1 < 0 && eq2 < 0)
        {
            eq1 = Math.abs(eq1);
            eq2 = Math.abs(eq2);
        }
        if(eq1 >= 0 && eq2 == -1){
            //when numrater is possitive and denominator is -
1 then inverse of -
1 will be 1 in EUA so numrater remains possitive that should not happen
            // to convert numrater to nagetive we multiply it with -
2 and add with itself
            eq1 = (eq1 * -2) + eq1;
        }


        long lemda = (eq1 * extendedEuclidian(eq2,prime)) % prime;
        R.x = (pow(lemda,2) - P.x - Q.x) % prime;
        R.x = R.x < 0 ? positiveInvers(R.x,prime) : R.x;
        R.y = (lemda*(P.x - R.x) - P.y) % prime;
        R.y = R.y < 0 ? positiveInvers(R.y,prime) : R.y;
        return R;
    }

    public static Point[] eccEncryption(Point M,Point e1,Point e2,long prime,l
ong a){

        System.out.println("\nEncryption :");
        long r = (int)Math.random() % prime;
        Point[] c = new Point[2];

        c[0]= keyGeneration(e1,r,prime,a);
        Point e1_r = keyGeneration(e2,r,prime,a);
        c[1] = pointsAddition(M,e1_r,prime);
```

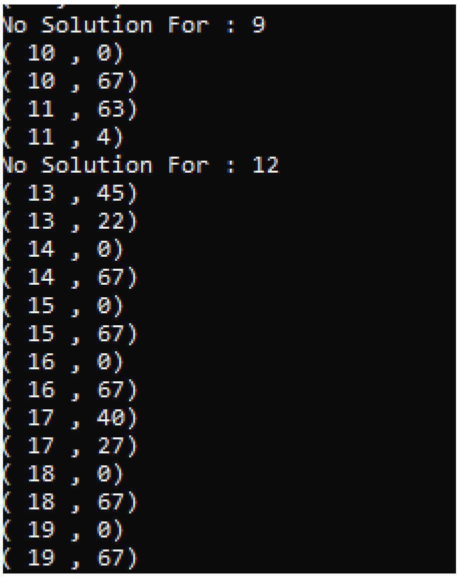


```
        System.out.println("c1 : x is : " + c[0].x + " y is :" + c[0].y);
        System.out.println("c2 : x is : " + c[1].x + " y is :" + c[1].y);

        return c;
    }


    public static void eccDesryption(Point[] c, long d,long prime,long a){
        System.out.println("Decryption :");
        // M = c2 - (d * c1)

        // d * c1
        Point d_c1 = keyGeneration(c[0],d,prime,a);

        // for subtraction operation we are giving negative sign to Y corrdina
te
        d_c1.y = (d_c1.y * -2) + d_c1.y;
        Point M = pointsAddition(c[1],d_c1,prime);
        System.out.println("M : x is : " + M.x + " y is :" + M.y);

    }
    public static void main(String[] args) {

        long p = 67;
        Point P = new Point(2,22);
        int i=0;
        long a=2,b=3;
        /*
        while(true){
            if(((int)Math.pow(i,3)*4 + 27*(int)Math.pow(i,2))!=0){
                a=i;
                b=i;
                break;
            }i++;
        }
       */
        long d=4;
        //assune P == e1
        elipticalCurve(a,b,p);
        Point e2 = keyGeneration(P,d,p,a);
        Point M = new Point(24,26);
        System.out.println("\ne1 is : (" + P.x + "," + P.y + ")");
        System.out.println("e2 is : (" + e2.x + "," + e2.y + ") \n d is : " +
d);
        System.out.println("Message is : " + "("+M.x+","+M.y+")");

        Point[] c = eccEncryption(M,P,e2,p,a);
        eccDesryption(c,d,p,a);
    }
}
```

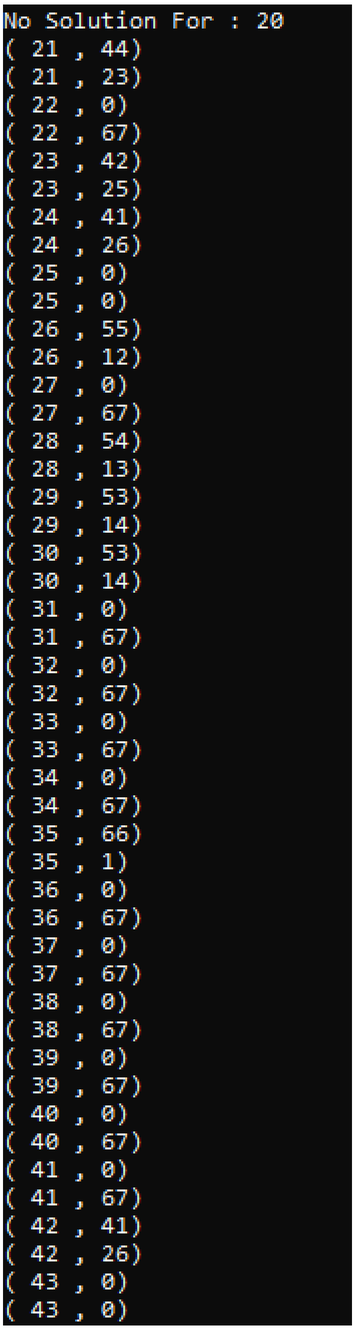
Output: -



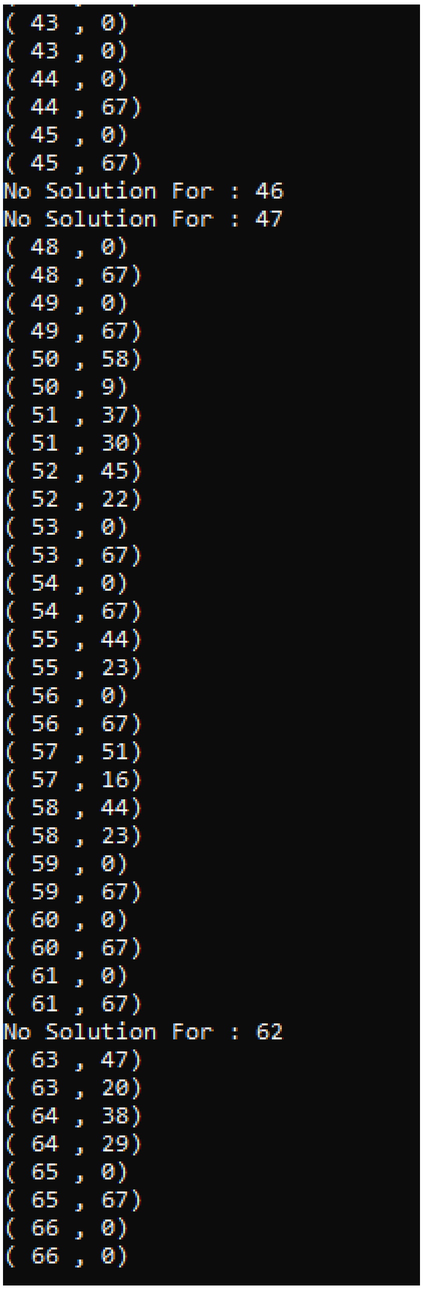
```
PS D:\DDIT\sem6\NIS\LAB\lab7> java ECC
No Solution For : 0
( 1 , 41)
( 1 , 26)
( 2 , 45)
( 2 , 22)
( 3 , 61)
( 3 , 6)
( 4 , 0)
( 4 , 67)
( 5 , 65)
( 5 , 2)
( 6 , 0)
( 6 , 67)
( 7 , 62)
( 7 , 5)
( 8 , 53)
( 8 , 14)
```



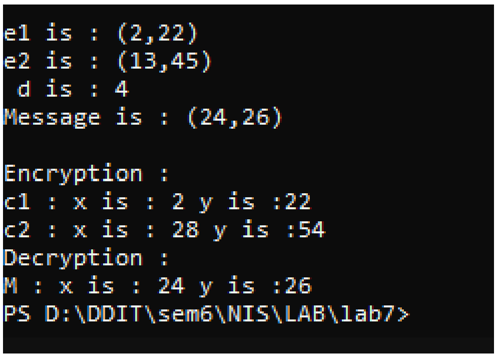
```
No Solution For : 9
( 10 , 0)
( 10 , 67)
( 11 , 63)
( 11 , 4)
No Solution For : 12
( 13 , 45)
( 13 , 22)
( 14 , 0)
( 14 , 67)
( 15 , 0)
( 15 , 67)
( 16 , 0)
( 16 , 67)
( 17 , 40)
( 17 , 27)
( 18 , 0)
( 18 , 67)
( 19 , 0)
( 19 , 67)
```



```
No Solution For : 20
( 21 , 44)
( 21 , 23)
( 22 , 0)
( 22 , 67)
( 23 , 42)
( 23 , 25)
( 24 , 41)
( 24 , 26)
( 25 , 0)
( 25 , 0)
( 26 , 55)
( 26 , 12)
( 27 , 0)
( 27 , 67)
( 28 , 54)
( 28 , 13)
( 29 , 53)
( 29 , 14)
( 30 , 53)
( 30 , 14)
( 31 , 0)
( 31 , 67)
( 32 , 0)
( 32 , 67)
( 33 , 0)
( 33 , 67)
( 34 , 0)
( 34 , 67)
( 35 , 66)
( 35 , 1)
( 36 , 0)
( 36 , 67)
( 37 , 0)
( 37 , 67)
( 38 , 0)
( 38 , 67)
( 39 , 0)
( 39 , 67)
( 40 , 0)
( 40 , 67)
( 41 , 0)
( 41 , 67)
( 42 , 41)
( 42 , 26)
( 43 , 0)
( 43 , 0)
```



```
( 43 , 0)
( 43 , 0)
( 44 , 0)
( 44 , 67)
( 45 , 0)
( 45 , 67)
No Solution For : 46
No Solution For : 47
( 48 , 0)
( 48 , 67)
( 49 , 0)
( 49 , 67)
( 50 , 58)
( 50 , 9)
( 51 , 37)
( 51 , 30)
( 52 , 45)
( 52 , 22)
( 53 , 0)
( 53 , 67)
( 54 , 0)
( 54 , 67)
( 55 , 44)
( 55 , 23)
( 56 , 0)
( 56 , 67)
( 57 , 51)
( 57 , 16)
( 58 , 44)
( 58 , 23)
( 59 , 0)
( 59 , 67)
( 60 , 0)
( 60 , 67)
( 61 , 0)
( 61 , 67)
No Solution For : 62
( 63 , 47)
( 63 , 20)
( 64 , 38)
( 64 , 29)
( 65 , 0)
( 65 , 67)
( 66 , 0)
( 66 , 0)
```



```
e1 is : (2,22)
e2 is : (13,45)
 d is : 4
Message is : (24,26)

Encryption :
c1 : x is : 2 y is :22
c2 : x is : 28 y is :54
Decryption :
M : x is : 24 y is :26
PS D:\DDIT\sem6\NIS\LAB\lab7>
```

- Same as above(copied from console)

```
PS D:\DDIT\sem6\NIS\LAB\lab7> javac ECC.java

PS D:\DDIT\sem6\NIS\LAB\lab7> java ECC
```

```
No Solution For : 0
( 1 , 41)
( 1 , 26)
( 2 , 45)
( 2 , 22)
( 3 , 61)
( 3 , 6)
( 4 , 0)
( 4 , 67)
( 5 , 65)
( 5 , 2)
( 6 , 0)
( 6 , 67)
( 7 , 62)
( 7 , 5)
( 8 , 53)
( 8 , 14)
No Solution For : 9
( 10 , 0)
( 10 , 67)
( 11 , 63)
( 11 , 4)
No Solution For : 12
( 13 , 45)
( 13 , 22)
( 14 , 0)
( 14 , 67)
( 15 , 0)
( 15 , 67)
( 16 , 0)
( 16 , 67)
( 17 , 40)
( 17 , 27)
( 18 , 0)
( 18 , 67)
( 19 , 0)
( 19 , 67)
No Solution For : 20
( 21 , 44)
( 21 , 23)
( 22 , 0)
( 22 , 67)
( 23 , 42)
( 23 , 25)
( 24 , 41)
( 24 , 26)
( 25 , 0)
( 25 , 0)
( 26 , 55)
( 26 , 12)
( 27 , 0)
( 27 , 67)
( 28 , 54)
( 28 , 13)
( 29 , 53)
( 29 , 14)
( 30 , 53)
( 30 , 14)
( 31 , 0)
( 31 , 67)
( 32 , 0)
( 32 , 67)
( 33 , 0)
( 33 , 67)
( 34 , 0)
( 34 , 67)
( 35 , 66)
( 35 , 1)
( 36 , 0)
( 36 , 67)
( 37 , 0)
( 37 , 67)
( 38 , 0)
( 38 , 67)
( 39 , 0)
( 39 , 67)
( 40 , 0)
( 40 , 67)
( 41 , 0)
( 41 , 67)
( 42 , 41)
( 42 , 26)
( 43 , 0)
( 43 , 0)
( 44 , 0)
( 44 , 67)
( 45 , 0)
( 45 , 67)
No Solution For : 46
No Solution For : 47
( 48 , 0)
( 48 , 67)
( 49 , 0)
( 49 , 67)
( 50 , 58)
( 50 , 9)
( 51 , 37)
( 51 , 30)
( 52 , 45)
( 52 , 22)
( 53 , 0)
( 53 , 67)
( 54 , 0)
( 54 , 67)
( 55 , 44)
( 55 , 23)
( 56 , 0)
( 56 , 67)
( 57 , 51)
( 57 , 16)
( 58 , 44)
( 58 , 23)
( 59 , 0)
( 59 , 67)
( 60 , 0)
( 60 , 67)
( 61 , 0)
( 61 , 67)
No Solution For : 62
( 63 , 47)
( 63 , 20)
( 64 , 38)
( 64 , 29)
( 65 , 0)
( 65 , 67)
( 66 , 0)
( 66 , 0)

e1 is : (2,22)
e2 is : (13,45)
 d is : 4
Message is : (24,26)

Encryption :
c1 : x is : 2 y is :22
c2 : x is : 28 y is :54
```

```
Decryption :

M : x is : 24 y is :26

PS D:\DDIT\sem6\NIS\LAB\lab7>
```