GLA UNIVERSITY
ESTD. 2010
ऋते ज्ञानान्न मुक्तिः

GLA
UNIVERSITY
MATHURA
Established vide U.P. Act 21 of 2010.

LAB FILE

Name: Arpit Verma

Semester/Year: 7th/ 4th year

University Roll no.: 191500151

Section: G (10)
Subject: Cryptography & Network Security(BCSE 0071)

Submitted to:

Dr. KAMAL NARAYAN KAMLESH

(BCSE 0071)

Objective➔1

Write a program to implement Additive Cipher(Z26)

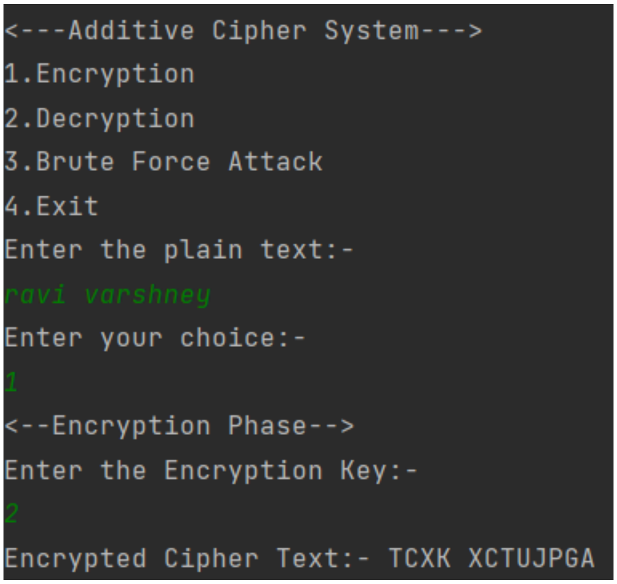
Code➔

```
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class AdditiveCipher {
    static ArrayList<String> enryptionList=new ArrayList<>();
    private static int bruteForceKeyFinder(char[] chars,String plain){
        HashMap<Integer,String> bruteforcemap=new HashMap<>();
        for(int i=1;i<=26;i++){
            String res=decryption(i,chars).trim();
            bruteforcemap.put(i,res);
        }
        int ans=0;
        for (int key: bruteforcemap.keySet()){
            String temp=bruteforcemap.get(key);
            if(plain.equals(temp)){
                ans=key;
            }
        }
        System.out.println(bruteforcemap);
        return ans;
    }
    private static String decryptNewCipher(int decryption_key,char []
chars,String [] arr){
        ArrayList<String> plainList=new ArrayList<>();
        String plaintext="";
        for(String i:arr){
            String convert=i.toLowerCase();
            String ans="";
            for(char ch:convert.toCharArray()){
                int temp=((ch-'a')-decryption_key)%26;
                if(temp<0){
                    temp+=26;
                    temp=temp%26;
                }
                else{
                    temp=temp%26;
                }
                ans+=chars[temp];
            }
            plainList.add(ans);
        }
        for(String i:plainList){
            plaintext+=i+" ";
        }
        return plaintext.trim();
    }
    private static String decryption(int decryption_key,char[] chars){
        ArrayList<String> plainList=new ArrayList<>();
        String plaintext="";
        for(String i:enryptionList){
         String convert=i.toLowerCase();
         String ans="";
         for(char ch:convert.toCharArray()){
            int temp=((ch-'a')-decryption_key)%26;
            if(temp<0){
               temp+=26;
               temp=temp%26;
            }
            else{
               temp=temp%26;
            }
            ans+=chars[temp];
         }
         plainList.add(ans);
      }
      for(String i:plainList){
         plaintext+=i+" ";
      }
      return plaintext.trim();
   }
   private static String encryption(String [] array,int encryption_key,char
[] chars){
      String encryptedCipher="";
      for(String i:array){
         ArrayList<Integer> clist=new ArrayList<>();
         for(char ch:i.toCharArray()){
            int temp=((ch-'a')+encryption_key)%26;
            clist.add(temp);
            }
            String s="";
            for(int j:clist){
                s+=chars[j];
            }
            enryptionList.add(s);
        }
        for(int i=0;i<enryptionList.size();i++){
            encryptedCipher+=enryptionList.get(i)+" ";
        }
        return encryptedCipher.trim();
    }
    private static boolean
checkForSpecialCharactersOtherThanSpaces(String [] array){
        int count=0;
        for(String i:array){
            if(isContainsOtherCharacters(i)){
                count++;
            }
        }
        return count>0;
    }
    private static boolean checkCipher(String [] array){
        int count=0;
        for(String i:array){
            if(isContainOtherThanUpperCase(i)){
                count++;
            }
        }
        return count>0;
    }
    private static void showDetails(){
        System.out.println("1.Encryption");
        System.out.println("2.Decryption");
        System.out.println("3.Brute Force Attack");
        System.out.println("4.Exit");
    }
    private static boolean isContainOtherThanUpperCase(String cipher){
        Pattern pat=Pattern.compile("[^A-Z]");
        Matcher mat= pat.matcher(cipher);
        return mat.find();
    }
    private static boolean isContainsOtherCharacters(String plain){
        Pattern pattern = Pattern.compile("[^a-z]");
        Matcher matcher = pattern.matcher(plain);
        return matcher.find();
    }
    private static boolean comparePlainWithCipherUsingKey(int key,char
[] chars,String plain){
        System.out.println("Decrypted Plain Text:-
"+decryption(key,chars));
        return plain.equals(decryption(key,chars));
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("<---Additive Cipher System--->");
```

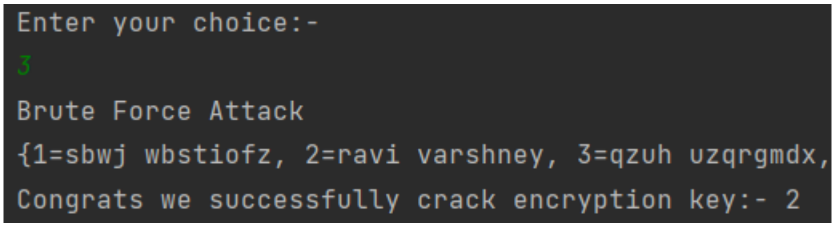
```
showDetails();
System.out.println("Enter the plain text:-");
String plain=sc.nextLine();
String [] arrayOfplain=plain.split(" ");
while(checkForSpecialCharactersOtherThanSpaces(arrayOfplain)){
    System.out.println("Re-enter Plain Text");
    plain=sc.nextLine();
    arrayOfplain=plain.split(" ");
}
System.out.println("Enter your choice:-");
int choice=sc.nextInt();
char [] chars=new char[26];
for(int i=0;i<26;i++){
    chars[i]= (char) ('a'+i);
}
String cipher="";
int key=0;
while(true){
    if(choice==1){
        System.out.println("<--Encryption Phase-->");
        System.out.println("Enter the Encryption Key:-");
        key=sc.nextInt();
        if(key>26){
            key=key%26;
        }
        cipher=encryption(arrayOfplain,key,chars).toUpperCase();
        System.out.println("Encrypted Cipher Text:- "+cipher);
        showDetails();
```

```
                System.out.println("Enter your choice:-");
                choice=sc.nextInt();
            }
            else if(choice==2){
                System.out.println("<--Decryption Started-->");
                System.out.println("Do you want to decrypt the previous
Cipher text,Enter 1 or 2");
                int ch=sc.nextInt();
                if(ch==1){
                    int decrypt_key=sc.nextInt();

if(comparePlainWithCipherUsingKey(decrypt_key,chars,plain)){
                        System.out.println("Plain Text Matched Successfully:-");
                    }
                    else {
                        System.out.println("No Match! Try Brute Force!!");
                    }
                }
                if(ch==2){
                    Scanner scn=new Scanner(System.in).useDelimiter("\n");
                    System.out.println("Enter CipherText:-");
                    String ci=scn.next();
                    String [] arr=ci.split(" ");
                    while(checkCipher(arr)){
                        System.out.println("Re-enter Cipher Text");
                        ci=scn.next();
                        arr=ci.split(" ");
                    }
                    System.out.println("Enter Decryption key:-");
                    int dkey=sc.nextInt();
                    String pl=decryptNewCipher(dkey,chars,arr);
                    if(plain.equals(pl)){
                        System.out.println("Congrats decryption successfully:-
"+pl);
                    }
                    else{
                        System.out.println("Try Brute Force.");
                    }
                }
                showDetails();
                System.out.println("Enter your choice:-");
                choice=sc.nextInt();
            }
            else if(choice==3){
                System.out.println("Brute Force Attack");
                int encrypted=bruteForceKeyFinder(chars,plain);
                System.out.println("Congrats we successfully crack encryption
key:- "+encrypted);
                System.out.println("Now Perform decryption to get plain
text.");
                System.out.println("Decryption Starting---->");
                System.out.println("Original Plain Text:-
"+decryption(encrypted,chars));
                showDetails();
                System.out.println("Enter your choice");
                choice=sc.nextInt();
            }
            else if(choice==4){
                System.exit(0);
            }
        }
    }
}
```

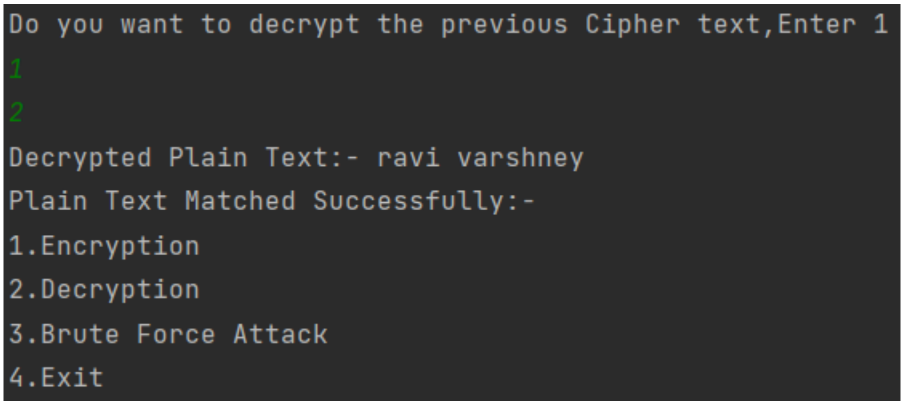
Screenshot➔



```
<---Additive Cipher System--->
1.Encryption
2.Decryption
3.Brute Force Attack
4.Exit
Enter the plain text:-
ravi varshney
Enter your choice:-
1
<--Encryption Phase-->
Enter the Encryption Key:-
2
Encrypted Cipher Text:- TCXK XCTUJPGA
```



```
Enter your choice:-
3
Brute Force Attack
{1=sbwj wbstiofz, 2=ravi varshney, 3=qzuh uzqrgmdx,
Congrats we successfully crack encryption key:- 2
```



```
Do you want to decrypt the previous Cipher text,Enter 1
1
2
Decrypted Plain Text:- ravi varshney
Plain Text Matched Successfully:-
1.Encryption
2.Decryption
3.Brute Force Attack
4.Exit
```

Objective➔2

Write a program to implement Multiplicative Cipher

Code➔

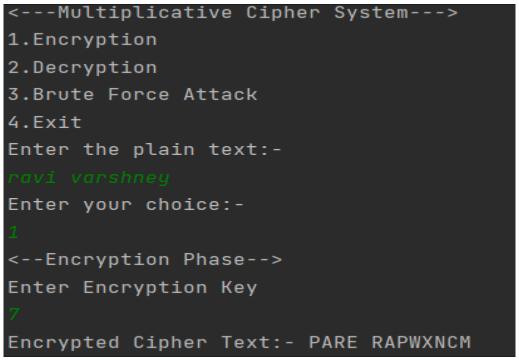
```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MultiplicativeCipher {
    private static String decryptNewCipher(int decryption_key,char []
chars,String [] arr){
        ArrayList<String> plainList=new ArrayList<>();
        String plainText="";
        for(String i:arr){
            String convert=i.toLowerCase();
            String ans="";
            for(char ch:convert.toCharArray()){
                int temp=((ch-'a')*map.get(decryption_key))%26;
                ans+=chars[temp];
            }
            plainList.add(ans);
        }
        for(String i:plainList){
            plainText+=i+" ";
    }
    return plainText.trim();
}
private static boolean checkCipher(String [] array){
    int count=0;
    for(String i:array){
        if(isContainOtherThanUpperCase(i)){
            count++;
        }
    }
    return count>0;
}
private static boolean isContainOtherThanUpperCase(String cipher){
    Pattern pat=Pattern.compile("[^A-Z]");
    Matcher mat= pat.matcher(cipher);
    return mat.find();
}
static ArrayList<String> enryptionList=new ArrayList<>();
static ArrayList<Integer> domain=new ArrayList<>();
private static int bruteForce(char [] chars,String plain){
    HashMap<Integer,String> bruteforcemap=new HashMap<>();
    for(int i:domain){
        String temp=decryption(i,chars).trim();
        bruteforcemap.put(i,temp);
    }
    int ans=0;
    for (int key: bruteforcemap.keySet()){
        String temp=bruteforcemap.get(key);
```

```
            if(plain.equals(temp)){
                ans=key;
            }
        }
        return ans;
    }
    private static String encryption(String [] array,int encryption_key,char
[] chars){
        String encryptedCipher="";
        for(String i:array){
            ArrayList<Integer> clist=new ArrayList<>();
            for(char ch:i.toCharArray()){
                int temp=((ch-'a')*encryption_key)%26;
                clist.add(temp);
            }
            String s="";
            for(int j:clist){
                s+=chars[j];
            }
            enryptionList.add(s);
        }
        for(int i=0;i<enryptionList.size();i++){
            encryptedCipher+=enryptionList.get(i)+" ";
        }
        return encryptedCipher.trim();
    }
    private static String decryption(int decryption_key,char[] chars){
        ArrayList<String> plainList=new ArrayList<>();
    String plaintext="";
    for(String i:enryptionList){
        String convert=i.toLowerCase();
        String ans="";
        for(char ch:convert.toCharArray()){
            int temp=((ch-'a')*map.get(decryption_key))%26;
            ans+=chars[temp];
        }
        plainList.add(ans);
    }
    for(String i:plainList){
        plaintext+=i+" ";
    }
    return plaintext.trim();
}

private static void showDetails(){
    System.out.println("1.Encryption");
    System.out.println("2.Decryption");
    System.out.println("3.Brute Force Attack");
    System.out.println("4.Exit");
}
private static boolean isContainsOtherCharacters(String plain){
    Pattern pattern = Pattern.compile("[^a-z]");
    Matcher matcher = pattern.matcher(plain);
    return matcher.find();
}
private static boolean
```

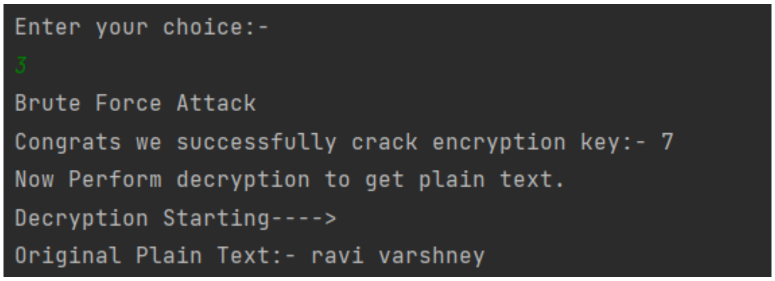
```
checkForSpecialCharactersOtherThanSpaces(String [] array){
        int count=0;
        for(String i:array){
            if(isContainsOtherCharacters(i)){
                count++;
            }
        }
        return count>0;
    }
    static HashMap<Integer,Integer> map=new HashMap<>();
    private static void modInverse(int a, int m)
    {
        for (int x = 1; x < m; x++)
            if (((a%m) * (x%m)) % m == 1)
                map.put(a,x);
    }
    private static boolean comparePlainWithCipherUsingKey(int key,char
[] chars,String plain){
        System.out.println("Decrypted Plain Text:-
"+decryption(key,chars));
        return plain.equals(decryption(key,chars));
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("<---Multiplicative Cipher System--->");
        showDetails();
        System.out.println("Enter the plain text:-");
        String plain=sc.nextLine();
String [] arrayOfplain=plain.split(" ");
while(checkForSpecialCharactersOtherThanSpaces(arrayOfplain)){
   System.out.println("Re-enter Plain Text");
   plain=sc.nextLine();
   arrayOfplain=plain.split(" ");
}
System.out.println("Enter your choice:-");
int choice=sc.nextInt();
char [] chars=new char[26];
for(int i=0;i<26;i++){
   chars[i]= (char) ('a'+i);
}
String cipher="";
for(int i=1;i<=26;i++){
   modInverse(i,26);
}
Set<Integer> keyDomain=map.keySet();
for(int i:keyDomain) domain.add(i);
while(true){
   if(choice==1){
      System.out.println("<--Encryption Phase-->");
      System.out.println("Enter Encryption Key");
      int encrypt_key=sc.nextInt();
      if(encrypt_key>26){
         encrypt_key=encrypt_key%26;
      }
      while(!keyDomain.contains(encrypt_key)){
         System.out.println("Please Enter valid key:-");
```

```
          encrypt_key=sc.nextInt();
        }

cipher=encryption(arrayOfplain,encrypt_key,chars).toUpperCase();
        System.out.println("Encrypted Cipher Text:- "+cipher);
        showDetails();
        System.out.println("Enter your choice:-");
        choice=sc.nextInt();

      }
      if(choice==2){
        System.out.println("<--Decryption Phase-->");
        System.out.println("Do you want to decrypt the previous
encrypted text enter choice 1 or 2");
        int ch=sc.nextInt();
        if(ch==1){
          System.out.println("Enter Decryption Key");
          int decrypt_key=sc.nextInt();

if(comparePlainWithCipherUsingKey(decrypt_key,chars,plain)){
            System.out.println("Plain Text Matched Successfully:-");
          }
          else {
            System.out.println("No Match! Try Brute Force!!");
          }
        }
        if(ch==2){
          Scanner scn=new Scanner(System.in).useDelimiter("\n");
          System.out.println("Enter CipherText:-");
          String ci=scn.next();
          String [] arr=ci.split(" ");
          while(checkCipher(arr)){
            System.out.println("Re-enter Cipher Text");
            ci=scn.next();
            arr=ci.split(" ");
          }
          System.out.println("Enter Decryption key:-");
          int dkey=sc.nextInt();
          String pl=decryptNewCipher(dkey,chars,arr);
          System.out.println(pl);
          if(plain.equals(pl)){
            System.out.println("Congrats decryption done
successfully:-"+pl);
          }
          else{
            System.out.println("Try Brute Force");
          }
        }
        showDetails();
        System.out.println("Enter your choice:-");
        choice=sc.nextInt();
      }
      if(choice==3){
        System.out.println("Brute Force Attack");
        int encrypted=bruteForce(chars,plain);
        System.out.println("Congrats we successfully crack encryption
key:- "+encrypted);
                System.out.println("Now Perform decryption to get plain
text.");
                System.out.println("Decryption Starting---->");
                System.out.println("Original Plain Text:-
"+decryption(encrypted,chars));
                showDetails();
                System.out.println("Enter your choice");
                choice=sc.nextInt();
            }
            if(choice==4){
                System.exit(0);
            }
        }
    }
}
```

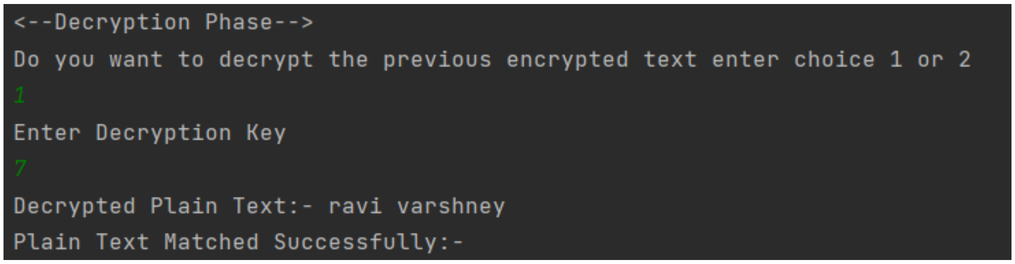
Screenshot➔



```
<---Multiplicative Cipher System--->
1.Encryption
2.Decryption
3.Brute Force Attack
4.Exit
Enter the plain text:-
ravi varshney
Enter your choice:-
1
<--Encryption Phase-->
Enter Encryption Key
7
Encrypted Cipher Text:- PARE RAPWXNCM
```



```
Enter your choice:-
3
Brute Force Attack
Congrats we successfully crack encryption key:- 7
Now Perform decryption to get plain text.
Decryption Starting---->
Original Plain Text:- ravi varshney
```



```
<--Decryption Phase-->
Do you want to decrypt the previous encrypted text enter choice 1 or 2
1
Enter Decryption Key
7
Decrypted Plain Text:- ravi varshney
Plain Text Matched Successfully:-
```

Objective➔3

Write a program to implement Affine Cipher.

Code➔

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class AffineCipher {
   private static final String alphabet = "abcdefghijklmnopqrstuvwxyz";
   private static final String alphabet1 =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in).useDelimiter("\n");
      List<Integer> list = new ArrayList<>();
      Collections.addAll(list, 1, 3, 5,7,11,15,17,19,21,23,25);

      System.out.println("1 FOR ENCRYPTION:");
      System.out.println("2 FOR DECRYPTION:");
      System.out.println("3 FOR BRUTEFORCE:");
      System.out.println("Other Key FOR EXIT:");
      int c=sc.nextInt();
      boolean result=false;
      String pt="";
String ct="";
String pt1="";
String ptd="";
String ct1="";
String km="";

String keym="";
String keya="";

int kmf=0;
String ka="";
int kaf=0;
int f=0;
ArrayList<Integer> spaces=new ArrayList<>();
switch(c){
    case 1:
        while(!result){
            System.out.println("ENTER Plaintext : ");
            pt=sc.next();
            pt1=pt;
            pt=pt.replaceAll("\\s+","");
            result = pt.matches("[a-z]+");
            if(result==false)
                System.out.println("ENTER CORRECT STRING::");
        }
        for(int i=0;i<pt1.length();i++){
            if(pt1.charAt(i)==' ') spaces.add(i);
        }
```

```
while (f == 0) {
    System.out.println("ENTER Multiplicative KEY : ");
    km = sc.next();
    result = km.matches("[0-9]+");
    if (result == false)
        System.out.println("ENTER CORRECT KEY::");
    else
        kmf = Integer.parseInt(km);
    kmf = kmf % 26;
    if(list.contains(kmf)) {
        f = 1;
    }
    else {
        System.out.println("ENTER CORRECT KEY::");
        System.out.println("Key Must Be::1,3,5,7,11,15,17,19,21,23,25");
        result=false;
    }
}
f=0;
while (f == 0) {
    System.out.println("ENTER Addative KEY : ");
    ka = sc.next();
    result = ka.matches("[0-9]+");
    if (result == false)
        System.out.println("ENTER CORRECT KEY::");
    else
        f = 1;
    }
    kaf= Integer.parseInt(ka);
    kaf = kaf % 26;

    String enc = autoAEncryption(pt,kmf,kaf);
    String enc1 = "";
    StringBuffer str = new StringBuffer(enc);

    for (int i = 0; i < pt1.length(); i++) {
        for (int j = 0; j < spaces.size(); j++) {
            if (spaces.get(j) == i) {
                str.insert(i, ' ');
            }
        }

    }
    enc1 = str.toString();
    System.out.println("Plaintext : " + pt1);
    System.out.println("Encrypted : " + enc1);
    break;


case 2:
    while(!result){
        System.out.println("ENTER Ciphertext : ");
        ptd=sc.next();
        pt1=ptd;
        ptd=ptd.replaceAll("\\s+","");
```

```
            result = ptd.matches("[A-Z]+");
            if(result==false)
                System.out.println("ENTER CORRECT STRING::");
        }
        for(int i=0;i<pt1.length();i++){
            if(pt1.charAt(i)==' ') spaces.add(i);
        }

        for(int i=0;i<ptd.length();i++){
            int p1=alphabet1.indexOf(ptd.charAt(i));
            pt+=alphabet.charAt(p1);
        }
        while (f == 0) {
            System.out.println("ENTER Multiplicative KEY : ");
            km = sc.next();
            result = km.matches("[0-9]+");
            if (result == false)
                System.out.println("ENTER CORRECT KEY::");
            else
                kmf = Integer.parseInt(km);
            kmf = kmf % 26;
            if(list.contains(kmf)) {
                f = 1;
            }
            else {
                System.out.println("ENTER CORRECT KEY::");
                System.out.println("Key Must
Be::1,3,5,7,11,15,17,19,21,23,25");
```

```
                        result=false;
                    }
                }
                f=0;
                while (f == 0) {
                    System.out.println("ENTER Addative KEY : ");
                    ka = sc.next();
                    result = ka.matches("[0-9]+");
                    if (result == false)
                        System.out.println("ENTER CORRECT KEY::");
                    else
                        f = 1;
                }
                kaf= Integer.parseInt(ka);
                kaf = kaf % 26;
                int counterInverse=1;
                while ((kmf*counterInverse)%26!=1){
                    counterInverse+=1;
                }
//               System.out.println(counterInverse);
//               System.out.println(kaf);
                String dec = autoADecryption(pt,counterInverse,kaf);
                String dec1 = "";
                StringBuffer str1 = new StringBuffer(dec);
//               System.out.println(spaces);
                for (int i = 0; i < pt1.length(); i++) {
                    for (int j = 0; j < spaces.size(); j++) {
                        if (spaces.get(j) == i) {
```

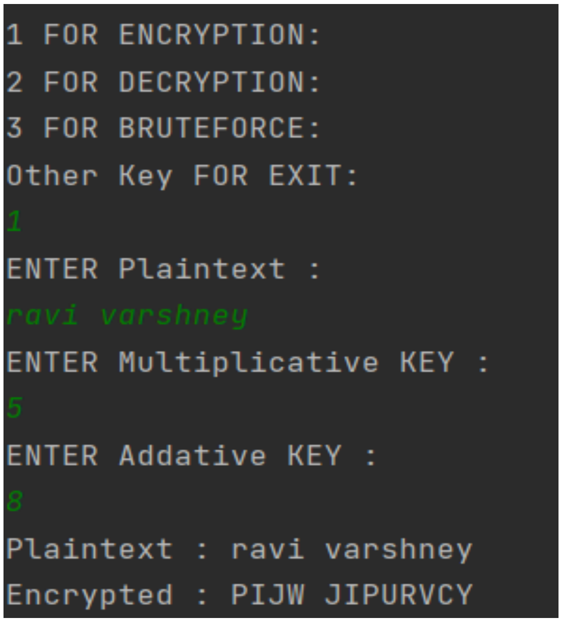
```
                str1.insert(i, ' ');
            }
        }

    }
    dec1 = str1.toString();
    System.out.println("Encrypted : " + pt1);
    System.out.println("Plaintext : " + dec1);
    break;

case 3:

    String ciptext = "";
    String ciptext1 = "";
    boolean ctresult = false;
    boolean outerloop = false;
    while (!outerloop) {
        result = false;
        ctresult = false;
        while (!result) {
            System.out.println("ENTER Plaintext : ");
            pt = sc.next();
            pt1 = pt;
            pt = pt.replaceAll("\\s+", "");
            result = pt.matches("[a-z]+");
            if (result == false)
                System.out.println("ENTER CORRECT STRING::");
        }
```

```
while (!ctresult) {
    System.out.println("ENTER Ciphertext : ");
    ciptext = sc.next();
    ciptext1 = ciptext;
    ciptext = ciptext.replaceAll("\\s+", "");
    ctresult = ciptext.matches("[A-Z]+");
    if (ctresult == false)
        System.out.println("ENTER CORRECT STRING::");
}


int flag = 1;
if (ciptext1.length() != pt1.length()) {
    System.out.println("Length of both are not same");
    flag = 0;
}
int flagC = 1;
if (flag == 1) {
    for (int i = 0; i < pt1.length(); i++) {
        if (pt1.charAt(i) == ' ') {
            if (ciptext1.charAt(i) == ' ') {

            } else {
                flagC = 0;
            }
        }
    }
```

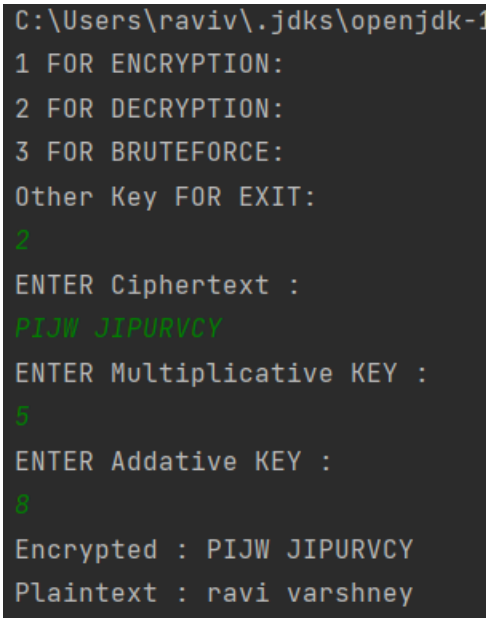
```
            }

            if (flagC == 0) {
                System.out.println("Spaces are not equal or at same
place");
            }
            if (flagC == 1 && flag == 1) {
                outerloop = true;
            }
        }



        String Encr = "";
        int keyBruteM = 0;
        int keyBruteA = 0;
        int flag1 = 0;
        for (int i = 0; i < 26; i++) {
            keyBruteA=i;
            for (int j = 0; j < 26; j++) {
                keyBruteM=j;
                Encr = autoAEncryption(pt, keyBruteM,keyBruteA);
                //System.out.println("KEY:"+i+"  "+Encr);
                if (Encr.equals(ciptext)) {
                    System.out.println("FOUND Key ADDITIVE:" + i);
                    System.out.println("FOUND Key MULTIPLICATIVE:" + j);
                    flag1 = 1;
                    break;
                }
```

```
                    Encr = "";
                    if(flag1==1) break;
                }
                if(flag1==1) break;
            }



            if (flag1 == 0)
                System.out.println("NO RESULT FOUND");


            break;
        }



    }


    public static String autoAEncryption(String msg,int km,int ka){
        int len = msg.length();
        String encryptMsg = "";
        for(int i=0;i<len;i++){

            int p1=alphabet.indexOf(msg.charAt(i));
            int total = ((p1*km)+ka)% 26;
            encryptMsg += alphabet1.charAt(total);
        }
        //System.out.println(encryptMsg);
        return encryptMsg;
    }


    public static String autoADecryption(String msg,int km,int ka)
    {
        int len = msg.length();
        String decryptMsg = "";
        for(int i=0;i<len;i++){
            int p1=alphabet.indexOf(msg.charAt(i));
            int total = ((p1-ka)*km)% 26;
            if(total<0){
                total+=26;
            }
            decryptMsg += alphabet.charAt(total);
        }
        //System.out.println(decryptMsg);
        return decryptMsg;
    }


}
```

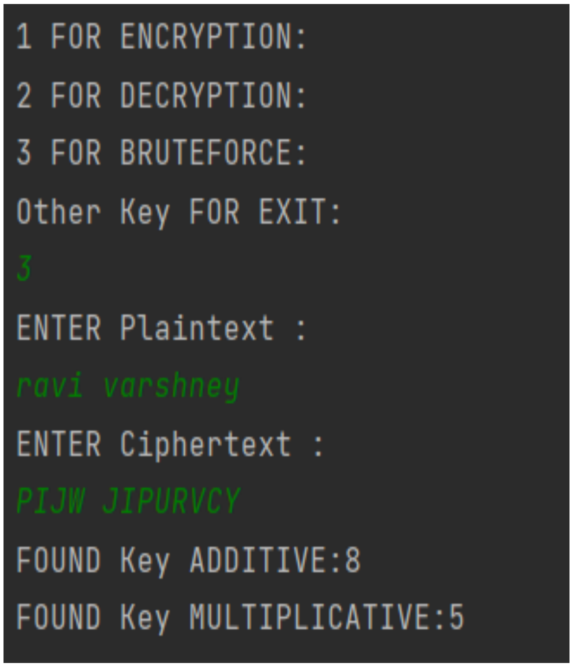
Screenshot➔



```
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
1
ENTER Plaintext :
ravi varshney
ENTER Multiplicative KEY :
5
ENTER Addative KEY :
8
Plaintext : ravi varshney
Encrypted : PIJW JIPURVCY
```



```
C:\Users\raviv\.jdks\openjdk-1
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
2
ENTER Ciphertext :
PIJW JIPURVCY
ENTER Multiplicative KEY :
5
ENTER Addative KEY :
8
Encrypted : PIJW JIPURVCY
Plaintext : ravi varshney
```



```
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
3
ENTER Plaintext :
ravi varshney
ENTER Ciphertext :
PIJW JIPURVCY
FOUND Key ADDITIVE:8
FOUND Key MULTIPLICATIVE:5
```

Objective➔4

Write a program to implement Vignere Cipher.

Code➔

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class VigenereCipher {
    private static final String alphabet = "abcdefghijklmnopqrstuvwxyz";
    private static final String alphabet1 =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in).useDelimiter("\n");
        List<Integer> list = new ArrayList<>();
        Collections.addAll(list, 1, 3, 5, 7, 11, 15, 17, 19, 21, 23, 25);

        System.out.println("1 FOR ENCRYPTION:");
        System.out.println("2 FOR DECRYPTION:");
        System.out.println("3 FOR BRUTEFORCE:");
        System.out.println("Other Key FOR EXIT:");
        int c = sc.nextInt();
        boolean result = false;
        String pt = "";
        String ct = "";
        String pt1 = "";
String ptd = "";
String ct1 = "";
String km = "";

String key = "";

int kmf = 0;
String ka = "";
int kaf = 0;
int f = 0;
ArrayList<Integer> spaces = new ArrayList<>();
switch (c) {
    case 1:
        while(!result){
            System.out.println("ENTER Plaintext : ");
            pt=sc.next();
            pt1=pt;
            pt=pt.replaceAll("\\s+","");
            result = pt.matches("[a-z]+");
            if(result==false)
                System.out.println("ENTER CORRECT STRING::");
        }
        for(int i=0;i<pt1.length();i++){
            if(pt1.charAt(i)==' ') spaces.add(i);
        }
        while (f == 0) {
            System.out.println("ENTER KEY : ");
            key = sc.next();
```

```
                result = key.matches("[a-z]+");
                if (result == false)
                    System.out.println("ENTER CORRECT KEY::");
                else
                    f = 1;
            }
            String keyL=generateKey(pt,key);
            String enc1=cipherText(pt,keyL);


            StringBuffer str1 = new StringBuffer(enc1);
//                 System.out.println(spaces);
            for (int i = 0; i < pt1.length(); i++) {
                for (int j = 0; j < spaces.size(); j++) {
                    if (spaces.get(j) == i) {
                        str1.insert(i, ' ');
                    }
                }

            }
            enc1 = str1.toString();
            System.out.println("Plaintext : " + pt1);
            System.out.println("Encrypted : " + enc1);
            break;



        case 2:
```

```
while(!result){
    System.out.println("ENTER Ciphertext : ");
    ptd=sc.next();
    pt1=ptd;
    ptd=ptd.replaceAll("\\s+","");
    result = ptd.matches("[A-Z]+");
    if(result==false)
        System.out.println("ENTER CORRECT STRING::");
}
for(int i=0;i<pt1.length();i++){
    if(pt1.charAt(i)==' ') spaces.add(i);
}

for(int i=0;i<ptd.length();i++){
    int p1=alphabet1.indexOf(ptd.charAt(i));
    pt+=alphabet.charAt(p1);
}
while (f == 0) {
    System.out.println("ENTER KEY : ");
    key = sc.next();
    result = key.matches("[a-z]+");
    if (result == false)
        System.out.println("ENTER CORRECT KEY::");
    else
        f = 1;
}
String keyE=generateKey(pt,key);
String dec1=originalText(pt,keyE);
```

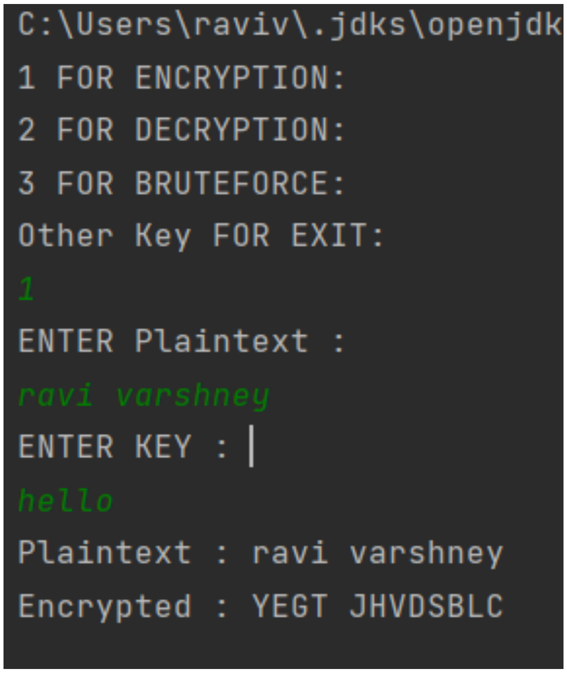
```
            StringBuffer str2 = new StringBuffer(dec1);
//                 System.out.println(spaces);
            for (int i = 0; i < pt1.length(); i++) {
                for (int j = 0; j < spaces.size(); j++) {
                    if (spaces.get(j) == i) {
                        str2.insert(i, ' ');
                    }
                }

            }
            dec1 = str2.toString();
            System.out.println("Plaintext : " + pt1);
            System.out.println("Encrypted : " + dec1);
            break;


        case 3:

            String ciptext = "";
            String ciptext1 = "";
            boolean ctresult = false;
            boolean outerloop = false;

            while (!outerloop) {
                result = false;
                ctresult = false;
```

```
while (!result) {
    System.out.println("ENTER Plaintext : ");
    pt = sc.next();
    pt1 = pt;
    pt = pt.replaceAll("\\s+", "");
    result = pt.matches("[a-z]+");
    if (result == false)
        System.out.println("ENTER CORRECT STRING::");
}

while (!ctresult) {
    System.out.println("ENTER Ciphertext : ");
    ptd = sc.next();
    ciptext1 = ptd;
    ptd = ptd.replaceAll("\\s+", "");
    ctresult = ptd.matches("[A-Z]+");
    if (ctresult == false)
        System.out.println("ENTER CORRECT STRING::");
}

for(int i=0;i<ptd.length();i++){
    int p33=alphabet1.indexOf(ptd.charAt(i));
    ciptext+=alphabet.charAt(p33);
}

int flag = 1;
if (ciptext1.length() != pt1.length()) {
    System.out.println("Length of both are not same");
```

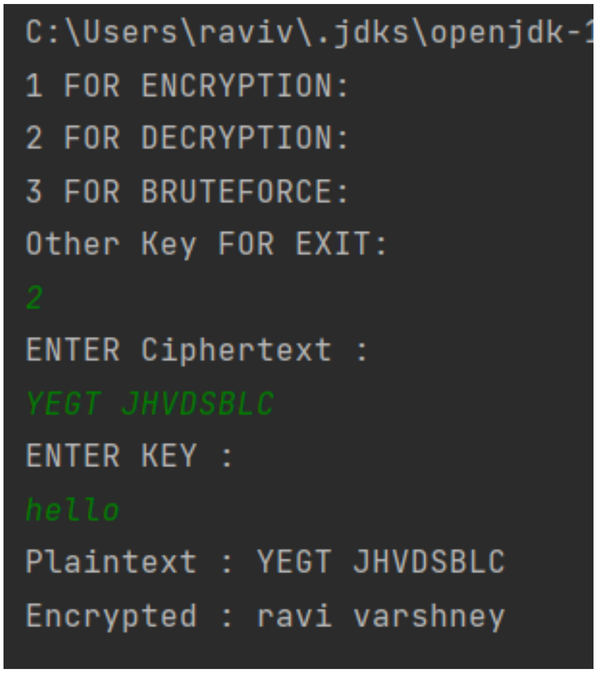
```
                        flag = 0;
                    }
                    int flagC = 1;
                    if (flag == 1) {
                        for (int i = 0; i < pt1.length(); i++) {
                            if (pt1.charAt(i) == ' ') {
                                if (ciptext1.charAt(i) == ' ') {

                                } else {
                                    flagC = 0;
                                }
                            }
                        }
                    }

                    if (flagC == 0) {
                        System.out.println("Spaces are not equal or at same
place");
                    }
                    if (flagC == 1 && flag == 1) {
                        outerloop = true;
                    }
                }



                String keyBrute = "";

                keyBrute=originalText(ciptext,pt);
```

```
            System.out.println("KEY : "+keyBrute);


            break;

    }
}



static String generateKey(String str, String key)
{
    int x = str.length();

    for (int i = 0; ; i++)
    {
        if (x == i)
            i = 0;
        if (key.length() == str.length())
            break;
        key+=(key.charAt(i));
    }
    return key;
}

static String cipherText(String str, String key)
{
    String cipher_text="";

    for (int i = 0; i < str.length(); i++)
    {
      int x = (alphabet.indexOf(str.charAt(i)) +
alphabet.indexOf(key.charAt(i))) %26;
      cipher_text+=alphabet1.charAt(x);
    }
    return cipher_text;
  }


  static String originalText(String cipher_text, String key)
  {
    String orig_text="";

    for (int i = 0 ; i < cipher_text.length() &&
        i < key.length(); i++)
    {
      int x = (alphabet.indexOf(cipher_text.charAt(i)) -
          alphabet.indexOf(key.charAt(i)) + 26) %26;
      orig_text+=alphabet.charAt(x);
    }
    return orig_text;
  }
}
```

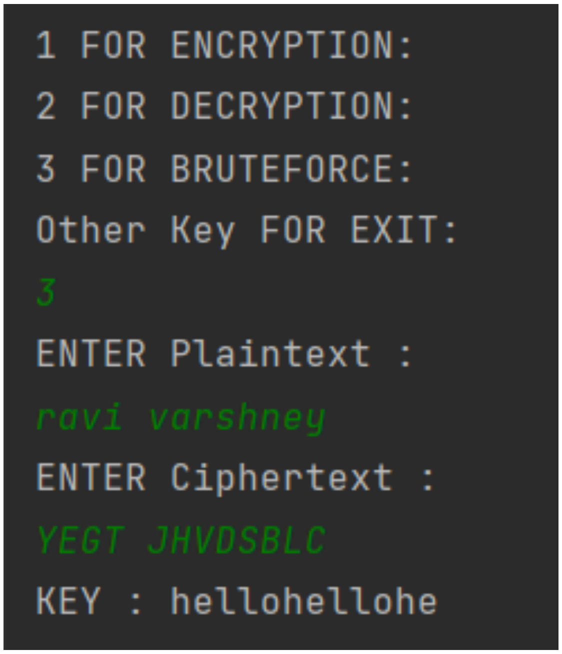
Screenshot→



```
C:\Users\raviv\.jdks\openjdk
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
1
ENTER Plaintext :
ravi varshney
ENTER KEY : |
hello
Plaintext : ravi varshney
Encrypted : YEGT JHVDSBLC
```



```
C:\Users\raviv\.jdks\openjdk-1
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
2
ENTER Ciphertext :
YEGT JHVDSBLC
ENTER KEY :
hello
Plaintext : YEGT JHVDSBLC
Encrypted : ravi varshney
```



```
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
3
ENTER Plaintext :
ravi varshney
ENTER Ciphertext :
YEGT JHVDSBLC
KEY : hellohellohe
```

Objective➔5

Write a program to implement Autokey Cipher.

Code➔

```
import java.security.Key;
import java.util.ArrayList;
import java.util.Locale;
import java.util.Optional;
import java.util.Scanner;

public class AutoKeyChiper {
    private static final String alphabet = "abcdefghijklmnopqrstuvwxyz";
    private static final String alphabet1 =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in).useDelimiter("\n");
        int loop1=1;
        while (loop1==1) {
            System.out.println("1 FOR ENCRYPTION:");
            System.out.println("2 FOR DECRYPTION:");
            System.out.println("3 FOR BRUTEFORCE:");
            System.out.println("Other Key FOR EXIT:");
            String key = "";
            String pt = "";
            String pt1 = "";
            String k11 = "";
            String ptd="";
            int f = 0;
            int k = 0;
            int c = sc.nextInt();
            boolean result = false;
            ArrayList<Integer> spaces = new ArrayList<>();

            switch (c) {
                case 1:


                    while (!result) {
                        System.out.println("ENTER Plaintext : ");
                        pt = sc.next();
                        pt1 = pt;
                        pt = pt.replaceAll("\\s+", "");
                        result = pt.matches("[a-z]+");
                        if (result == false)
                            System.out.println("ENTER CORRECT STRING::");
                    }
                    for (int i = 0; i < pt1.length(); i++) {
                        if (pt1.charAt(i) == ' ') spaces.add(i);
                    }
                    while (f == 0) {
                        System.out.println("ENTER KEY : ");
                        k11 = sc.next();
```

```
                    result = k11.matches("[0-9]+");
                    if (result == false)
                        System.out.println("ENTER CORRECT KEY::");
                    else
                        f = 1;
                }


                k = Integer.parseInt(k11);
                k = k % 26;

                key += alphabet.charAt(k);


                String enc = autoEncryption(pt, key);
                String enc1 = "";
                int cc = 0;
                StringBuffer str = new StringBuffer(enc);
//                 System.out.println(spaces);
                for (int i = 0; i < pt1.length(); i++) {
                    for (int j = 0; j < spaces.size(); j++) {
                        if (spaces.get(j) == i) {
                            str.insert(i, ' ');
                        }
                    }

                }
                enc1 = str.toString();
```

```
    System.out.println("Plaintext : " + pt1);
    System.out.println("Encrypted : " + enc1);
    break;


case 2:
    boolean result11 = false;
    while (!result11) {
        System.out.println("ENTER Encrypted : ");
        ptd = sc.next();
        pt1 = ptd;
        ptd = ptd.replaceAll("\\s+", "");
        result11 = ptd.matches("[A-Z]+");
        if (result11 == false)
            System.out.println("ENTER CORRECT STRING En::");
    }
    for (int i = 0; i < pt1.length(); i++) {
        if (pt1.charAt(i) == ' ') spaces.add(i);
    }

    for(int i=0;i<ptd.length();i++){
        int p1=alphabet1.indexOf(ptd.charAt(i));
        pt+=alphabet.charAt(p1);
    }

    while (f == 0) {
        System.out.println("ENTER KEY : ");
        k11 = sc.next();
```

```
                        result11 = k11.matches("[0-9]+");
                        if (result11 == false)
                            System.out.println("ENTER CORRECT KEY::");
                        else
                            f = 1;
                    }


                    k = Integer.parseInt(k11);
                    k = k % 26;
                    key += alphabet.charAt(k);
                    String dec = autoDecryption(pt, key);
                    StringBuffer str1 = new StringBuffer(dec);
//                     System.out.println(spaces);
                    for (int i = 0; i < pt1.length(); i++) {
                        for (int j = 0; j < spaces.size(); j++) {
                            if (spaces.get(j) == i) {
                                str1.insert(i, ' ');
                            }
                        }

                    }
                    enc1 = str1.toString();
                    System.out.println("Encrypted : " + pt1);
                    System.out.println("Decrypted : " + enc1);
                    break;

                case 3:
```

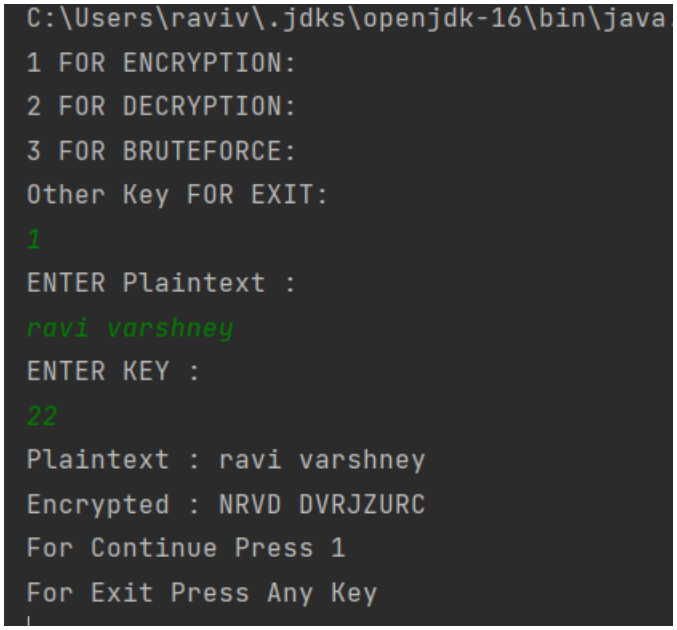
```
String ciptext = "";
String ciptext1 = "";
boolean ctresult = false;
boolean outerloop = false;
while (!outerloop) {
    result = false;
    ctresult = false;
    while (!result) {
        System.out.println("ENTER Plaintext : ");
        pt = sc.next();
        pt1 = pt;
        pt = pt.replaceAll("\\s+", "");
        result = pt.matches("[a-z]+");
        if (result == false)
            System.out.println("ENTER CORRECT STRING::");
    }

    while (!ctresult) {
        System.out.println("ENTER Ciphertext : ");
        ciptext = sc.next();
        ciptext1 = ciptext;
        ciptext = ciptext.replaceAll("\\s+", "");
        ctresult = ciptext.matches("[A-Z]+");
        if (ctresult == false)
            System.out.println("ENTER CORRECT STRING::");
    }
```

```
                        int flag = 1;
                        if (ciptext1.length() != pt1.length()) {
                            System.out.println("Length of both are not same");
                            flag = 0;
                        }
                        int flagC = 1;
                        if (flag == 1) {
                            for (int i = 0; i < pt1.length(); i++) {
                                if (pt1.charAt(i) == ' ') {
                                    if (ciptext1.charAt(i) == ' ') {

                                    } else {
                                        flagC = 0;
                                    }
                                }
                            }
                        }

                        if (flagC == 0) {
                            System.out.println("Spaces are not equal or at same
place");
                        }
                        if (flagC == 1 && flag == 1) {
                            outerloop = true;
                        }
                    }
```

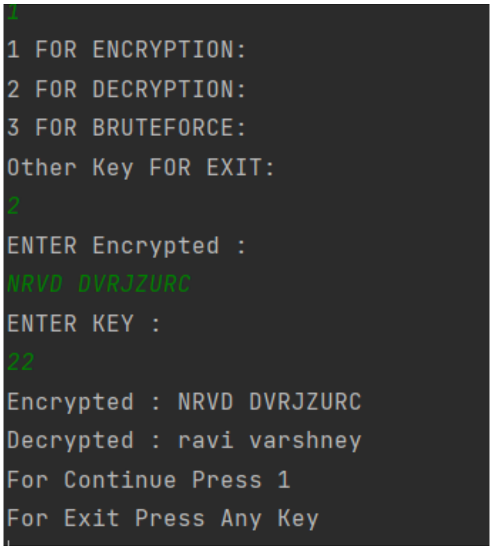
```
        String Encr = "";
        String keyBrute = "";
        int flag1 = 0;
        for (int i = 0; i < 26; i++) {
            keyBrute += alphabet.charAt(i);
            Encr = autoEncryption(pt, keyBrute);
            //System.out.println("KEY:"+i+"  "+Encr);
            if (Encr.equals(ciptext)) {
                System.out.println("FOUND Key:" + i);
                flag1 = 1;
                break;
            }
            keyBrute = "";
            Encr = "";
        }
        if (flag1 == 0)
            System.out.println("NO RESULT FOUND");


        break;


    default:
        System.out.println("EXIT:");
}
System.out.println("For Continue Press 1");
System.out.println("For Exit Press Any Key");
```

```
        loop1=sc.nextInt();
    }








}

public static String autoEncryption(String msg, String key)
{
    int len = msg.length();
    String newkey=key.concat(msg);
    newkey=newkey.substring(0,newkey.length()-key.length());
    String encryptMsg = "";
    for(int i=0;i<len;i++){
        int p1=alphabet.indexOf(msg.charAt(i));
        int n1=alphabet.indexOf(newkey.charAt(i));
        int total = (p1 + n1) % 26;
        encryptMsg += alphabet1.charAt(total);
    }
    return encryptMsg;
}


public static String autoDecryption(String msg, String key)
{
        String currentKey = key;
        String decryptMsg = "";
        for (int x = 0; x < msg.length(); x++) {
            int get1 = alphabet.indexOf(msg.charAt(x));
            int get2 = alphabet.indexOf(currentKey.charAt(x));
            int total = (get1 - get2) % 26;
            total = (total < 0) ? total + 26 : total;
            decryptMsg += alphabet.charAt(total);
            currentKey += alphabet.charAt(total);
        }
        return decryptMsg;
    }
}
```

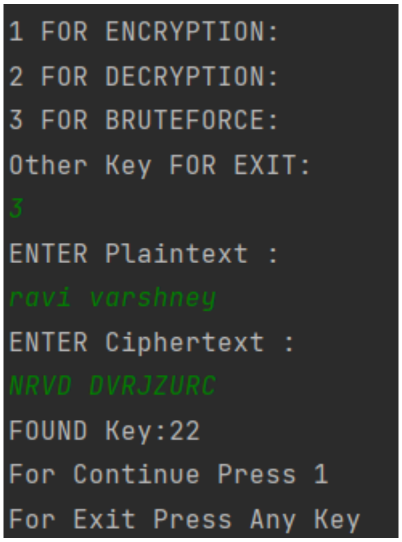
Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\java
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
1
ENTER Plaintext :
ravi varshney
ENTER KEY :
22
Plaintext : ravi varshney
Encrypted : NRVD DVRJZURC
For Continue Press 1
For Exit Press Any Key
```



```
1
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
2
ENTER Encrypted :
NRVD DVRJZURC
ENTER KEY :
22
Encrypted : NRVD DVRJZURC
Decrypted : ravi varshney
For Continue Press 1
For Exit Press Any Key
```



```
1 FOR ENCRYPTION:
2 FOR DECRYPTION:
3 FOR BRUTEFORCE:
Other Key FOR EXIT:
3
ENTER Plaintext :
ravi varshney
ENTER Ciphertext :
NRVD DVRJZURC
FOUND Key:22
For Continue Press 1
For Exit Press Any Key
```

Objective-6➔

Write a program to implement Columner Transposition technique.

Code➔

```
public class TranspositionCipher
{
   public static String selectedKey;
   public static char   sortedKey[];
   public static int    sortedKeyPos[];

   public TranspositionCipher()
```

```
{
    selectedKey = "megabuck";
    sortedKeyPos = new int[selectedKey.length()];
    sortedKey = selectedKey.toCharArray();
}

public TranspositionCipher(String myKey)
{
    selectedKey = myKey;
    sortedKeyPos = new int[selectedKey.length()];
    sortedKey = selectedKey.toCharArray();
}


public static void doProcessOnKey()
{

    int min, i, j;
    char orginalKey[] = selectedKey.toCharArray();
    char temp;
    // First Sort the array of selected key
    for (i = 0; i < selectedKey.length(); i++)
    {
        min = i;
        for (j = i; j < selectedKey.length(); j++)
        {
            if (sortedKey[min] > sortedKey[j])
            {
                min = j;
            }
        }
        if (min != i)
        {
            temp = sortedKey[i];
            sortedKey[i] = sortedKey[min];
            sortedKey[min] = temp;
        }
    }

    for (i = 0; i < selectedKey.length(); i++)
    {
        for (j = 0; j < selectedKey.length(); j++)
        {
            if (orginalKey[i] == sortedKey[j])
                sortedKeyPos[i] = j;
        }
    }
}

public static String doEncryption(String plainText)
{
    int min, i, j;
    char orginalKey[] = selectedKey.toCharArray();
    char temp;
    doProcessOnKey();
```

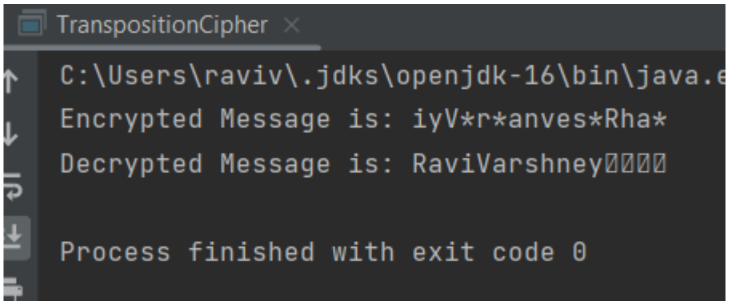
```
int row = plainText.length() / selectedKey.length();
int extrabit = plainText.length() % selectedKey.length();
int exrow = (extrabit == 0) ? 0 : 1;
int rowtemp = -1, coltemp = -1;
int totallen = (row + exrow) * selectedKey.length();
char pmat[][] = new char[(row + exrow)][(selectedKey.length())];
char encry[] = new char[totallen];
int tempcnt = -1;
row = 0;
for (i = 0; i < totallen; i++)
{
    coltemp++;
    if (i < plainText.length())
    {
        if (coltemp == (selectedKey.length()))
        {
            row++;
            coltemp = 0;
        }
        pmat[row][coltemp] = plainText.charAt(i);
    }
    else
    { // do the padding ...
        pmat[row][coltemp] = '*';
    }
}
int len = -1, k;
for (i = 0; i < selectedKey.length(); i++)
```

```
    {
        for (k = 0; k < selectedKey.length(); k++)
        {
            if (i == sortedKeyPos[k])
            {
                break;
            }
        }
        for (j = 0; j <= row; j++)
        {
            len++;
            encry[len] = pmat[j][k];
        }
    }
    String p1 = new String(encry);
    return (new String(p1));
}


public static String doDecryption(String s)
{
    int min, i, j, k;
    char key[] = selectedKey.toCharArray();
    char encry[] = s.toCharArray();
    char temp;
    doProcessOnKey();
    // Now generating plain message
    int row = s.length() / selectedKey.length();
```

```
char pmat[][] = new char[row][(selectedKey.length())];
int tempcnt = -1;
for (i = 0; i < selectedKey.length(); i++)
{
    for (k = 0; k < selectedKey.length(); k++)
    {
        if (i == sortedKeyPos[k])
        {
            break;
        }
    }
    for (j = 0; j < row; j++)
    {
        tempcnt++;
        pmat[j][k] = encry[tempcnt];
    }
}

char p1[] = new char[row * selectedKey.length()];
k = 0;
for (i = 0; i < row; i++)
{
    for (j = 0; j < selectedKey.length(); j++)
    {
        if (pmat[i][j] != '*')
        {
            p1[k++] = pmat[i][j];
        }
```

```
            }
        }
        p1[k++] = '\0';
        return (new String(p1));
    }

    @SuppressWarnings("static-access")
    public static void main(String[] args)
    {
        TranspositionCipher tc = new TranspositionCipher();
        System.out.println("Encrypted Message is: "
                + tc.doEncryption("RaviVarshney"));
        System.out.println("Decrypted Message is: "
                + tc.doDecryption(tc.doEncryption("RaviVarshney")));
    }
}
```

Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\java.e
Encrypted Message is: iyV*r*anves*Rha*
Decrypted Message is: RaviVarshney▯▯▯▯

Process finished with exit code 0
```

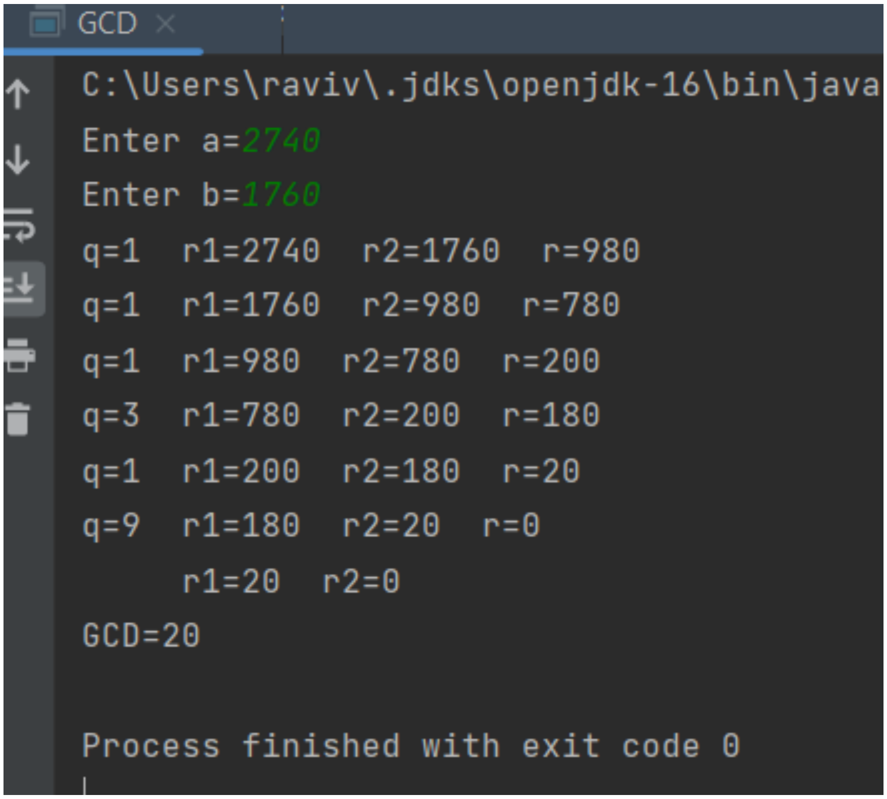
Objective-7➔

Write a program to implement Euclidean Algorithm to find GCD of given numbers.

Code➔

```
import java.util.Scanner;

public class GCD {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter a=");
        int a=sc.nextInt();
        System.out.print("Enter b=");
        int b=sc.nextInt();
        int r1=a;
        int r2=b;
        int r=0;
        int q=0;
        while (r2>0){
            q=r1/r2;
            r=r1-(q*r2);
            System.out.print("q="+q+"  r1="+r1+"  r2="+r2+"  r="+r);
            System.out.println();
            r1=r2;
            r2=r;
        }
        System.out.print("     r1="+r1+"  r2="+r2);
        System.out.println();
        System.out.println("GCD="+r1);}
}
```

Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\java
Enter a=2740
Enter b=1760
q=1  r1=2740  r2=1760  r=980
q=1  r1=1760  r2=980  r=780
q=1  r1=980  r2=780  r=200
q=3  r1=780  r2=200  r=180
q=1  r1=200  r2=180  r=20
q=9  r1=180  r2=20  r=0
     r1=20  r2=0
GCD=20

Process finished with exit code 0
```

Objective-8➔

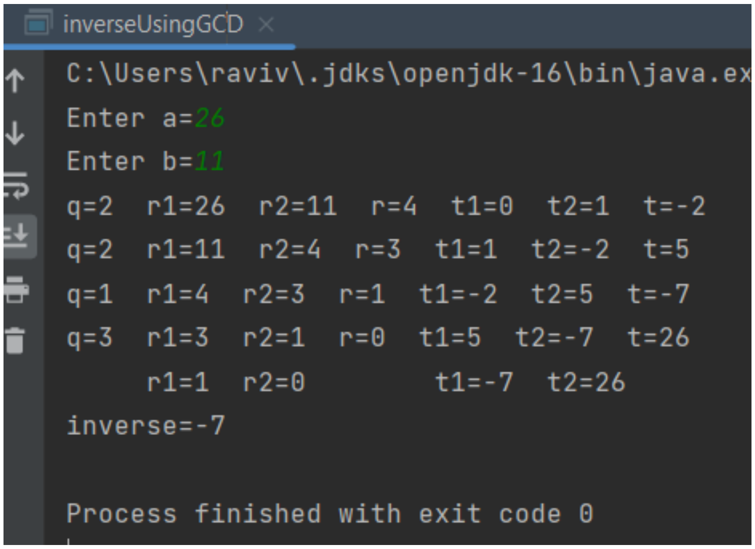
Write a program to find out the Multiplicative inverse of a given number using Extended Euclidean Algorithm.

Code➔

```
import java.util.Scanner;
public class inverseUsingGCD {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter a=");
        int a=sc.nextInt();
        System.out.print("Enter b=");
        int b=sc.nextInt();
        int r1=a;
        int r2=b;
        int r=0;
        int q=0;
        int t1=0;
        int t2=1;
        int t=0;
        while (r2>0){
            q=r1/r2;
            r=r1-(q*r2);
            t=t1-(q*t2);
            System.out.print("q="+q+"  r1="+r1+"  r2="+r2+"  r="+r+"
t1="+t1+"  t2="+t2+"  t="+t);
            System.out.println();
            t1=t2;
            t2=t;
            r1=r2;
            r2=r;
        }
        System.out.print("     r1="+r1+"  r2="+r2+"       t1="+t1+"  t2="+t2);
        System.out.println();
        if(r1==1)
        System.out.println("inverse="+t1);
        else
            System.out.println("Inverse does not exist");
```

```
    }
}
```

Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\java.ex
Enter a=26
Enter b=11
q=2  r1=26  r2=11  r=4  t1=0  t2=1  t=-2
q=2  r1=11  r2=4  r=3  t1=1  t2=-2  t=5
q=1  r1=4  r2=3  r=1  t1=-2  t2=5  t=-7
q=3  r1=3  r2=1  r=0  t1=5  t2=-7  t=26
     r1=1  r2=0       t1=-7  t2=26
inverse=-7

Process finished with exit code 0
```

Objective-9➔

Write a program to implement Elgamal Cryptosystem.

Code➔

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Scanner;
```

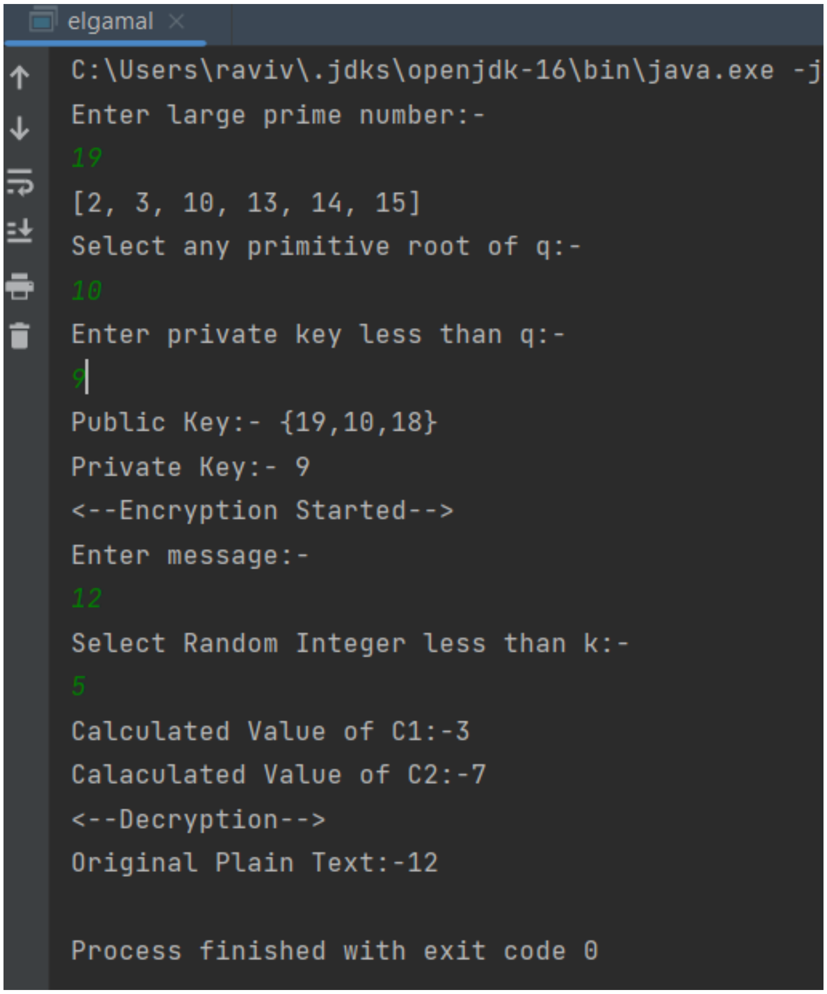
```
public class elgamal {
    static ArrayList<Integer> primitiveRootList=new ArrayList<>();
    static int power(int x, int y, int p)
    {
        int res = 1;

        x = x % p;
        if (x == 0)
            return 0;

        while (y > 0)
        {
            if ((y & 1) != 0)
                res = (res * x) % p;
            y = y >> 1;
            x = (x * x) % p;
        }
        return res;
    }
    static boolean isPrime(int n)
    {
        if (n <= 1)
        {
            return false;
        }
        if (n <= 3)
        {
```

```
        return true;
    }
    if (n % 2 == 0 || n % 3 == 0)
    {
        return false;
    }

    for (int i = 5; i * i <= n; i = i + 6)
    {
        if (n % i == 0 || n % (i + 2) == 0)
        {
            return false;
        }
    }

    return true;
}
static void findPrimefactors(List<Integer> s, int n)
{
    while (n % 2 == 0)
    {
        s.add(2);
        n = n / 2;
    }
    for (int i = 3; i <= Math.sqrt(n); i = i + 2)
    {
        while (n % i == 0)
        {
                s.add(i);
                n = n / i;
            }
        }
        if (n > 2)
        {
            s.add(n);
        }
    }
    private static int calculatePhi(int n){
        if(isPrime(n)) return n-1;
        List<Integer> listofPrime=new ArrayList<>();
        findPrimefactors(listofPrime,n);
        HashMap<Integer,Integer> map=new HashMap<>();
        for(Integer i:listofPrime){
            if(!map.containsKey(i)){
                map.put(i,1);
            }
            else{
                map.put(i,map.get(i)+1);
            }
        }
        int pro=1;
        for(int i:map.keySet()){
            if(map.get(i)>1){
                int calc=(int)Math.abs(Math.pow(i,map.get(i))-
Math.pow(i,map.get(i)-1));
                pro*=calc;
            }
            else{
                pro*=(i-1);
            }
        }
        return pro;
    }
    static void primtiveRootTable(int n){
        int phi=calculatePhi(n);
        int [][] matrix=new int[n+1][n+1];
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                int calc=power(i,j,n);
                matrix[i][j]=calc;
            }
        }
        HashMap<Integer,Integer> orderOfElement=new HashMap<>();
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                if(matrix[i][j]==1){
                    orderOfElement.put(i,j);
                    break;
                }
            }
        }
        for(Integer ele:orderOfElement.keySet()){

            if(orderOfElement.get(ele)==phi){
```

```
            primitiveRootList.add(ele);
        }
    }
    System.out.println(primitiveRootList);
}
static int modInverse(int A, int M)
{

    for (int X = 1; X < M; X++)
        if (((A % M) * (X % M)) % M == 1)
            return X;
    return 1;
}
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter large prime number:-");
    int q=sc.nextInt();
    primtiveRootTable(q);
    System.out.println("Select any primitive root of q:-");
    int alpha=sc.nextInt();
    System.out.println("Enter private key less than q:-");
    int xa=sc.nextInt();
    int ya=power(alpha,xa,q);
    System.out.println("Public Key:- "+"{"+q+","+alpha+","+ya+"}");
    System.out.println("Private Key:- "+xa);
    System.out.println("<--Encryption Started-->");
    System.out.println("Enter message:-");
    int m=sc.nextInt();
```

```
        System.out.println("Select Random Integer less than k:-");
        int k=sc.nextInt();
        int K=power(ya,k,q);
        int c1=power(alpha,k,q);
        int c2=K*m%q;
        System.out.println("Calculated Value of C1:-"+c1);
        System.out.println("Calaculated Value of C2:-"+c2);
        System.out.println("<--Decryption-->");
        int kval=power(c1,xa,q);
        int plain=((c2%q)*modInverse(K,q))%q;
        System.out.println("Original Plain Text:-"+plain);
    }
}
```

Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\java.exe -j
Enter large prime number:-
19
[2, 3, 10, 13, 14, 15]
Select any primitive root of q:-
10
Enter private key less than q:-
9
Public Key:- {19,10,18}
Private Key:- 9
<--Encryption Started-->
Enter message:-
12
Select Random Integer less than k:-
5
Calculated Value of C1:-3
Calaculated Value of C2:-7
<--Decryption-->
Original Plain Text:-12

Process finished with exit code 0
```

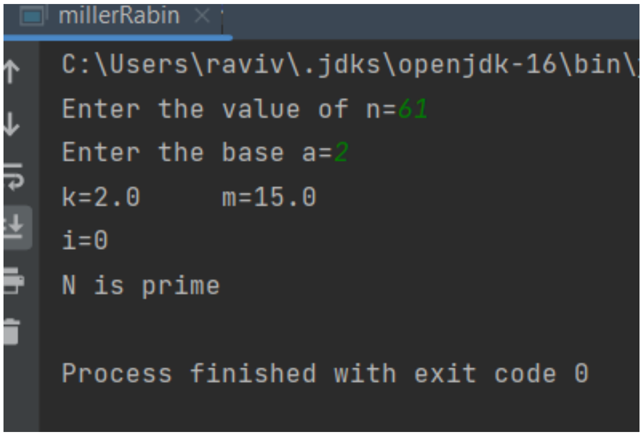
Objective-10➔

Write a program to implement Rabin Miller Primality Test to check given number is prime or composite.

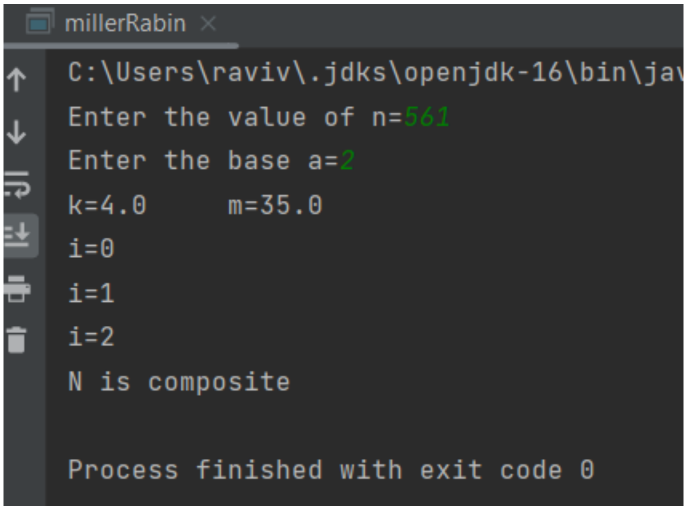
Code➔

```
import java.util.Scanner;

public class millerRabin {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the value of n=");
        int n=sc.nextInt();
        //System.out.println();
        System.out.print("Enter the base a=");
        double a=sc.nextInt();
        //System.out.println();
        double k=0,m=0;
        int temp=n-1;
        while (temp%2==0){
            k=k+1;
            temp=temp/2;
        }
        m=temp;
        int f=0;
        System.out.println("k="+k+"     m="+m);
        double b=Math.pow(a,m)%n;
        if(b==1){
            System.out.println("Prime");
        }
        else {
            for (int i = 0; i < k; i++) {
                System.out.println("i="+i);
                b=(b*b)%n;
                if(b==n-1) {
                    System.out.println("N is prime");
                    f=1;
                    break;
                }
                else if(b==1){
                    System.out.println("N is composite");
                    f=1;
                    break;
                }

            }
            if(f==0){
                System.out.println("N is composite");
            }
        }
    }
}
```

Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\
Enter the value of n=61
Enter the base a=2
k=2.0     m=15.0
i=0
N is prime

Process finished with exit code 0
```



```
C:\Users\raviv\.jdks\openjdk-16\bin\ja
Enter the value of n=561
Enter the base a=2
k=4.0     m=35.0
i=0
i=1
i=2
N is composite

Process finished with exit code 0
```

Objective-11➔

Write a program to implement RSA Algorithm.

Code➔

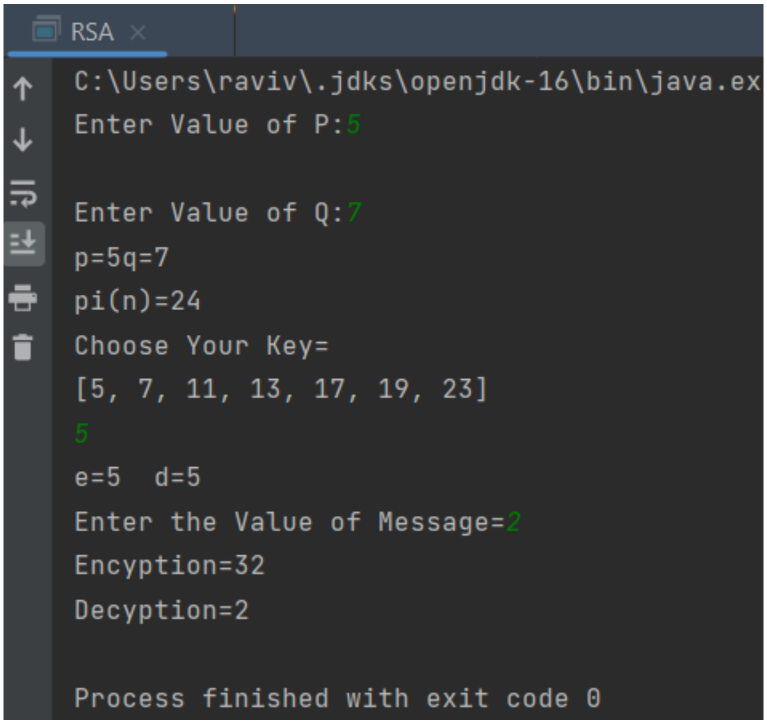
```
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class RSA {
    static List<Integer> d=new ArrayList<>();
    static List<Integer> e=new ArrayList<>();
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        boolean pF = false;
        boolean qF = false;
        boolean FF = false;
        int p = 0, q = 0;

        while (FF == false) {
            while (pF == false) {
                System.out.print("Enter Value of P:");
                p = sc.nextInt();
                pF = isPrime(p);
                if (pF == false)
                    System.out.println("Enter Prime Number");
            }
            System.out.println();
            while (qF == false) {
                System.out.print("Enter Value of Q:");
                q = sc.nextInt();
                qF = isPrime(q);
                if (qF == false)
                    System.out.println("Enter Prime Number");
            }

            if (p == q) {
                System.out.println("Both p and q are equal..Enter diffent
value");
                qF = false;
                pF = false;
                FF = false;
            } else {
                FF = true;
            }
        }

        System.out.println("p=" + p + "q=" + q);



        int n = p * q;
        int pn = (p - 1) * (q - 1);
        for (int i = 2; i <= pn; i++) {
            if (gcd(i, pn) == 1) {
```

```
        e.add(i);
    }
}
System.out.println("pi(n)="+pn);




boolean newFlag = false;

int k = 0;
while (newFlag == false) {
    System.out.println("Choose Your Key=");
    System.out.println(e);
    k = sc.nextInt();
    if (e.contains(k)) {
        newFlag = true;
    }
}
int dekey =0;

boolean aa=false;
while (aa==false){
    if((k*dekey)%pn==1){
        aa=true;
    }
    else dekey+=1;
}
```

```
   System.out.println("e="+k+"  d="+dekey);

   System.out.print("Enter the Value of Message=");
   double m = sc.nextInt();


   // Encryption c = (msg ^ e) % n
   long c = (long) (Math.pow(m,k))%n;
   System.out.println("Encyption="+c);


   // Decryption m = (c ^ d) % n
   BigInteger answer = BigInteger.valueOf(c).pow(dekey);
   BigInteger nn=BigInteger.valueOf(n);
   BigInteger ans=answer.mod(nn);
   System.out.println("Decyption="+ans);


}


public static boolean isPrime(int n)
{
   if (n <= 1)
      return false;
   for (int i = 2; i < n; i++)
```

```
            if (n % i == 0)
                return false;
        return true;
    }


    public static int gcd(int a, int h)
    {
        int temp;
        while (true)
        {
            temp = a%h;
            if (temp == 0)
                return h;
            a = h;
            h = temp;
        }
    }
}
```

Screenshot➔



```
C:\Users\raviv\.jdks\openjdk-16\bin\java.ex
Enter Value of P:5

Enter Value of Q:7
p=5q=7
pi(n)=24
Choose Your Key=
[5, 7, 11, 13, 17, 19, 23]
5
e=5  d=5
Enter the Value of Message=2
Encyption=32
Decyption=2

Process finished with exit code 0
```