Name :- Abhiu Garg
Section :- A1(03)
University Roll No. :- 191500042
Btech(CSE) Final Year

ALL LAB ASSIGNMENT

1). Implementation of Additive Cipher :

```
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);
        }
    }
  }
}
```

2) Implementation of Affine Cipher :

```
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 AffineCipher {
   static HashMap<ArrayList<Integer>,String> hmap=new HashMap<>();
   private static ArrayList<Integer> bruteForceKeyFinder(String [] cipherArray,char [] chars,String plain){
      for(int i:map.keySet()){
         int [] arr=new int[2];
         for(int j=1;j<=26;j++){
            arr[0]=i;
            arr[1]=j;
            ArrayList<Integer> lst=new ArrayList<>();
            for(int k:arr){
               lst.add(k);
            }
            String ans=decryption(i,j,cipherArray,chars);
            hmap.put(lst,ans);
         }
      }
      ArrayList<Integer> ans=new ArrayList<>();
      for(ArrayList<Integer> key:hmap.keySet()){
         if(plain.equals(hmap.get(key))){
            ans=key;
         }
      }
      return ans;
   }
   private static String decryption(int mkey,int akey,String [] cipherarray,char [] chars){
      String decrypted="";
      ArrayList<String> clist=new ArrayList<>();
      for(String i:cipherarray){
         String convert=i.toLowerCase();
         String ans="";
         ArrayList<Integer> list=new ArrayList<>();
         for(char ch:convert.toCharArray()){
            int temp=((ch-'a')-akey);
            if(temp<0){
               temp+=26;
            }
            int val=(temp*map.get(mkey))%26;
```

```
            ans+=chars[val];
        }
        clist.add(ans);
    }
    for(String i:clist){
        decrypted+=i+" ";
    }
    return decrypted.trim();
}
private static String encryption(int mkey,int akey,String [] array,char [] chars){
    ArrayList<String> plist=new ArrayList<>();
    String encrypted="";
    for(String i:array){
        ArrayList<Integer> elist=new ArrayList<>();
        for(char ch:i.toCharArray()){
            int temp=(((ch-'a')*mkey)+akey)%26;
            elist.add(temp);
        }
        String ans="";
        for(int j:elist) ans+=chars[j];
        plist.add(ans);
    }
    for(String i:plist){
        encrypted+=i+" ";
    }
    return encrypted.trim();
}
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 void showDetails(){
    System.out.println("1.Encryption");
    System.out.println("2.Decryption");
    System.out.println("3.Brute Force Attack");
    System.out.println("4.Exit");
}
static ArrayList<Integer> domain=new ArrayList<>();
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;
}
```

```
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();
}
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("<---Affine 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);
    String mkey;
    String akey;
    int kl=0;
    while(true){
        if(choice==1){
            System.out.println("Encryption");
            System.out.println("Enter multiplicative encryption key:-");
            mkey=sc.next();
            boolean result = mkey.matches("[0-9]+");
            while(!result) {
                System.out.println("Enter correct key");
                mkey = sc.next();
                result = mkey.matches("[0-9]+");
            }
            int ky=Integer.parseInt(mkey);
            while(!keyDomain.contains(ky)){
                System.out.println("Enter valid key from valid key domain:-");
```

```
            System.out.println(keyDomain);
            mkey=sc.next();
            ky = Integer.parseInt(mkey);
        }
        kl=ky;
        if(ky>26){
            ky=ky%26;
        }
        System.out.println("Enter additive key for encryption:-");
        akey=sc.next();
        boolean res = akey.matches("[0-9]+");
        while(!res){
            System.out.println("Enter correct key");
            akey=sc.next();
            res=akey.matches("[0-9]+");
        }
        int ak=Integer.parseInt(akey);
        if(ak>26) ak=ak%26;
        cipher=encryption(ky,ak,arrayOfplain,chars);
        System.out.println("Cipher Text:-"+cipher.toUpperCase());
        System.out.println("Enter choice");
        choice=sc.nextInt();
    }
    if(choice==2){
        System.out.println("Decryption");
        System.out.println("do you want to decrypt the previous one or new one enter 1 or 2:-");
        int ch=sc.nextInt();
        if(ch==1){
            System.out.println("Enter Multiplicative key:-");
            String mk=sc.next();
            boolean result = mk.matches("[0-9]+");
            while(!result){
                System.out.println("Enter correct key");
                mk=sc.next();
                result=mk.matches("[0-9]+");
            }
            int ky=Integer.parseInt(mk);
            kl=ky;
            if(ky>26){
                ky=ky%26;
            }
            System.out.println("Enter additive key for encryption:-");
            String aky=sc.next();
            boolean res = aky.matches("[0-9]+");
            while(!res){
                System.out.println("Enter correct key");
                aky=sc.next();
                res=aky.matches("[0-9]+");
            }
            int ak=Integer.parseInt(aky);
            if(ak>26) ak=ak%26;
            String [] arrayofcipher=cipher.split(" ");
            String decrypted=decryption(ky,ak,arrayofcipher,chars);
            System.out.println("Decrypted Text:- "+decrypted);
        }
```

```
            if(ch==2){
                Scanner scn=new Scanner(System.in).useDelimiter("\n");
                System.out.println("Enter Cipher Text");
                String ci=scn.next();
                String [] array=ci.split(" ");
                while (checkCipher(array)){
                    System.out.println("Re -enter Cipher Text");
                    ci=scn.next();
                    array=ci.split(" ");
                }
                System.out.println("Enter Multiplicative key:-");
                String mk=sc.next();
                boolean result = mk.matches("[0-9]+");
                while(!result){
                    System.out.println("Enter correct key");
                    mk=sc.next();
                    result=mk.matches("[0-9]+");
                }
                int ky=Integer.parseInt(mk);
                kl=ky;
                if(ky>26){
                    ky=ky%26;
                }
                System.out.println("Enter additive key for encryption:-");
                String aky=sc.next();
                boolean res = aky.matches("[0-9]+");
                while(!res){
                    System.out.println("Enter correct key");
                    aky=sc.next();
                    res=aky.matches("[0-9]+");
                }
                int ak=Integer.parseInt(aky);
                if(ak>26) ak=ak%26;
                String newDecrypted=decryption(ky,ak,array,chars);
                System.out.println("Decryted Text:- "+newDecrypted);
            }
            System.out.println("Next choice");
            choice=sc.nextInt();
        }
        if(choice==3){
            System.out.println("Brute Force");
            ArrayList<Integer> ans=bruteForceKeyFinder(cipher.split(" "),chars,plain);
            System.out.println("Multiplicative key is:- "+ans.get(0));
            System.out.println("Additive Key is:- "+ans.get(1));;
            System.out.println("Enter choice");
            choice=sc.nextInt();
        }
        if(choice==4){
            System.exit(0);
        }
    }
  }
}
```

3) Implementation of AutoKey Cipher :

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

public class AutoKeyCipher {
   static ArrayList<String> elist=new ArrayList<>();
   static ArrayList<ArrayList<Integer>> keyList=new ArrayList<>();
   static ArrayList<ArrayList<Integer>> dkeyList=new ArrayList<>();
   private static boolean isContainOtherThanUpperCase(String cipher){
      Pattern pat=Pattern.compile("[^A-Z]");
      Matcher mat= pat.matcher(cipher);
      return mat.find();
   }
   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");
   }
   static HashMap<Integer,String> map=new HashMap<>();
   private static int bruteForceKeyFinder(String plain,char [] chars,String [] array){
      for(int i=1;i<=26;i++){
         ArrayList<ArrayList<Integer>> plist=brutefill(array,i);
         String val=bruteDecrypter(plist,array,chars);
         map.put(i,val);
      }
      int ans=0;
      for(int i:map.keySet()){
         if(plain.equals(map.get(i))){
            ans=i;
         }
      }
      return ans;
   }
   private static ArrayList<ArrayList<Integer>> brutefill(String [] array,int key){
      ArrayList<ArrayList<Integer>> list=new ArrayList<>();
      for(String i:array){
         ArrayList<Integer> klist=new ArrayList<>();
         int [] karr=new int[i.length()];
```

```
        karr[0]=key;
        String convert=i.toLowerCase();
        for(int j=1;j<i.length();j++){
            int tep=(convert.charAt(j-1)-'a')-karr[j-1];
            if(tep<0){
                tep+=26;
            }
            karr[j]=tep;
        }
        for(int item:karr){
            klist.add(item);
        }
        list.add(klist);
        int p=convert.charAt(convert.length()-1)-'a'-karr[karr.length-1];
        if(p<0){
            p+=26;
        }
        key=p;
    }
    return list;
}
private static String bruteDecrypter(ArrayList<ArrayList<Integer>> lst,String[] array,char [] chars){
    String decrypted="";
    ArrayList<String> plist=new ArrayList<>();
    for(int i=0;i<lst.size();i++){
        String ans="";
        String str=array[i].toLowerCase();
        for(int j=0;j<lst.get(i).size();j++){
            int temp=((str.charAt(j)-'a')-lst.get(i).get(j));
            if(temp<0){
                temp+=26;
                temp=temp%26;
            }
            else{
                temp=temp%26;
            }
            ans+=chars[temp];
        }
        plist.add(ans);
    }
    for(String i:plist){
        decrypted+=i+" ";
    }
    return decrypted.trim();
}
private static String decryptnewCipher(String [] array,char [] chars){
    String decrypted="";
    ArrayList<String> plist=new ArrayList<>();
    for(int i=0;i<dkeyList.size();i++){
        String ans="";
        String str=array[i].toLowerCase();
        for(int j=0;j<dkeyList.get(i).size();j++){
            int temp=((str.charAt(j)-'a')-dkeyList.get(i).get(j));
            if(temp<0){
                temp+=26;
```

```
            temp=temp%26;
          }
          else{
            temp=temp%26;
          }
          ans+=chars[temp];
        }
        plist.add(ans);
    }
    for(String i:plist){
      decrypted+=i+" ";
    }
    return decrypted.trim();
}
private static String  decryption(char [] chars){
    String decrypted="";
    ArrayList<String> plist=new ArrayList<>();
    for(int i=0;i<keyList.size();i++){
      String ans="";
      String str=elist.get(i).toLowerCase();
      for(int j=0;j<keyList.get(i).size();j++){
        int temp=((str.charAt(j)-'a')-keyList.get(i).get(j));
        if(temp<0){
          temp+=26;
          temp=temp%26;
        }
        else{
          temp=temp%26;
        }
        ans+=chars[temp];
      }
      plist.add(ans);
    }
    for(String i:plist){
      decrypted+=i+" ";
    }
    return decrypted.trim();
}
private static boolean checkForSpecialCharactersOtherThanSpaces(String [] array){
    int count=0;
    for(String i:array){
      if(isContainsOtherCharacters(i)){
        count++;
      }
    }
    return count>0;
}
private static boolean isContainsOtherCharacters(String plain){
    Pattern pattern = Pattern.compile("[^a-z]");
    Matcher matcher = pattern.matcher(plain);
    return matcher.find();
}
private static void keyListFiller(String [] array,int key){
    for(String i:array){
      ArrayList<Integer> klist=new ArrayList<>();
```

```
        int [] karr=new int[i.length()];
        karr[0]=key;
        String convert=i.toLowerCase();
        for(int j=1;j<i.length();j++){
            int tep=(convert.charAt(j-1)-'a')-karr[j-1];
            if(tep<0){
                tep+=26;
            }
            karr[j]=tep;
        }
        for(int item:karr){
            klist.add(item);
        }
        dkeyList.add(klist);
        int p=convert.charAt(convert.length()-1)-'a'-karr[karr.length-1];
        if(p<0){
            p+=26;
        }
        key=p;
    }
}
private static String encryption(String [] array,int encryption_key,char [] chars){
    String encryptedsCipher="";

    for(String i:array){
        ArrayList<Integer> klist=new ArrayList<>();
        klist.add(encryption_key);
        String substr=i.substring(0,i.length()-1);
        int k=(i.charAt(i.length()-1)-'a');
        for(char c:substr.toCharArray()){
            int temp=c-'a';
            klist.add(temp);
        }
        keyList.add(klist);
        String s="";
        for(int j=0;j<i.length();j++){
            int t=((i.charAt(j)-'a')+klist.get(j))%26;
            s+=chars[t];
        }
        elist.add(s);
        encryption_key=k;
    }
    for(String i:elist){
        encryptedsCipher+=i+" ";
    }
    return encryptedsCipher.trim();
}
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Auto-Key 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="";
String key;
int kl=0;
while (true){
    if(choice==1){
        System.out.println("<--Encryption Phase-->");
        System.out.println("Enter Encryption Key:-");
        key=sc.next();
        boolean result = key.matches("[0-9]+");
        while(!result){
            System.out.println("Enter correct key");
            key=sc.next();
            result=key.matches("[0-9]+");
        }
        int ky=Integer.parseInt(key);
        kl=ky;
        if(ky>26){
            ky=ky%26;
        }
        cipher=encryption(arrayOfplain,ky,chars);
        System.out.println("Encrypted Cipher Text:- "+cipher.toUpperCase());
        System.out.println("Enter your choice for next step:-");
        choice=sc.nextInt();
    }
    if(choice==2){
        System.out.println("<--Decryption Started-->");
        System.out.println("Do you want to decrypt the previous text enter 1 or 2");
        int ch=sc.nextInt();
        if(ch==1){
            System.out.println("Enter decryption key");
            String decryt_key=sc.next();
            boolean varl=decryt_key.matches("[0-9]+");
            while(!varl){
                System.out.println("RE-enter key");
                decryt_key=sc.next();
                varl=decryt_key.matches("[0-9]+");
            }
            int dk=Integer.parseInt(decryt_key);
            if(plain.equals(decryption(chars)) && kl==dk){
                System.out.println("Congrats decryption done successfully "+decryption(chars));
            }
            else{
                System.out.println("Try Brute Force");
            }
        }
```

```
            if(ch==2){
               Scanner scn=new Scanner(System.in).useDelimiter("\n");
               System.out.println("Enter Cipher Text");
               String ct=scn.next();
               String [] arr=ct.split(" ");
               while(checkCipher(arr)){
                  System.out.println("Re-enter Cipher text");
                  ct=scn.next();
                  arr=ct.split(" ");
               }
               String [] newArray=new String[arr.length];
               for(int i=0;i<arr.length;i++){
                  newArray[i]=arr[i];
               }
               System.out.println("Enter Decryption key:-");
               String dkey=sc.next();
               boolean res=dkey.matches("[0-9]+");
               while(!res){
                  System.out.println("Re-enter key");
                  dkey=sc.next();
                  res=dkey.matches("[0-9]+");
               }
               int wk=Integer.parseInt(dkey);
               keyListFiller(newArray,wk);
               if(plain.equals(decryptnewCipher(newArray,chars))){
                  System.out.println("Yeah you find the original message in one go:- "+decryptnewCipher(newArray,c
hars));
               }
               else{
                  System.out.println("New Decrypted text:- "+decryptnewCipher(newArray,chars));
               }

            }
            System.out.println("Enter choice for next step");
            choice=sc.nextInt();
         }
         if(choice==3){
            System.out.println("<--Brute Force Approach-->");
            int keybf=bruteForceKeyFinder(plain,chars,cipher.split(" "));
            System.out.println("Yeah We find the hidden encrypted key:-"+keybf);
            System.out.println("Enter choice");
            choice=sc.nextInt();
         }
         if(choice==4){
            System.exit(0);
         }
      }

   }
}
```

4) DiffieHellmanExchangeALgorithm :

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

public class DiffieHellmanExchangeALgo {
    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);
}
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter prime number:- ");
    int q= sc.nextInt();;
    while (!isPrime(q)){
        System.out.println("Re-enter prime:- ");
        q= sc.nextInt();
    }
    primtiveRootTable(q);
    System.out.println("Select primitive root of q:- ");
    int alpha=sc.nextInt();
    while(!primitiveRootList.contains(alpha)){
        System.out.println("Re-enter primitive root:-");
        alpha=sc.nextInt();
    }
    System.out.println("Enter private key xa:- ");
    int xa=sc.nextInt();
    int ya=power(alpha,xa,q);
    System.out.println("Enter private key xb:- ");
    int xb=sc.nextInt();
    int yb=power(alpha,xb,q);
    System.out.println("Value of ya:- "+ya);
    System.out.println("Value of yb:- "+yb);
    int kab1=power(yb,xa,q);
    int kab2=power(ya,xb,q);
    System.out.println("Secret Key User A:- "+kab1);
    System.out.println("Secret Key User B:- "+kab2);
    if(kab1==kab2){
        System.out.println("Secret Shared Key:- "+kab1);
    }
    else{
        System.out.println("No Match");
    }
}
}
```

5) ElgamalCryptosystem :

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

```
import java.util.Scanner;

public class ElgamalCryptosystem {
   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:-");
        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);
    }
}
```

6) ElgamalSignatureScheme :

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

public class ElgamalSignatureScheme {
```

```
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;
}
static int gcd(int num1,int num2){
    if(num1<num2){
        int temp=num1;
        num1=num2;
        num2=temp;
    }
    int r1=num1;
    int r2=num2;
    int q;
    int r;
    while(r2>0){
        q=r1/r2;
        r=r1-q*r2;
        r1=r2;
        r2=r;
    }
    return r1;
}
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter prime number:-");
    int q=sc.nextInt();
    while(!isPrime(q)){
        System.out.println("Enter prime number:-");
        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("Enter msg:-");
    int msg=sc.nextInt();

    List<Integer> listofrandomvalues=new ArrayList<>();
    for (int i=1;i<=q-1;i++){
        if(gcd(i,q-1)==1) listofrandomvalues.add(i);
    }
    System.out.println("Random Integer List having gcd 1:-");
    System.out.println(listofrandomvalues);
```

```
        System.out.println("Select random integer k");
        int k=sc.nextInt();
        int s1=power(alpha,k,q);
        int kinv=modInverse(k,q-1);
        int cal=((msg-xa*s1)*kinv)%(q-1);
        if(cal<0){
            cal+=(q-1);
        }
        System.out.println("Signature S1:-"+s1);
        System.out.println("Signature S2:-"+cal);
        System.out.println("Now time for verification:-");
        int v1=power(alpha,msg,q);
        System.out.println("Value of v1:-"+v1);
        int v2=(power(ya,s1,q)*power(s1,cal,q))%q;
        System.out.println("Value of v2:-"+v2);
        if(v1==v2){
            System.out.println("Verified Successfuly:-");
        }
        else{
            System.out.println("Not Verified Successfully:-");
        }
    }
}
```

7) EucledianAlgorithm :

```
import java.util.Scanner;

public class EucledianAlgorithm {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int num1=sc.nextInt();
        int num2=sc.nextInt();
        if(num1<num2){
            int temp=num1;
            num1=num2;
            num2=temp;
        }
        int r1=num1;
        int r2=num2;
        int q;
        int r;
        while(r2>0){
            q=r1/r2;
            r=r1-q*r2;
            r1=r2;
            r2=r;
        }
        System.out.println("GCD of Two Numbers:- "+r1);
    }
}
```

8) Finding Multiplicative Inverse using ExtendedEucledianAlgorithm :

```
import java.util.Scanner;

public class ExtendedEucledianAlgorithm {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter num:-");
      int num=sc.nextInt();
      System.out.println("Enter modulous:-");
      int mod=sc.nextInt();
      if(num<mod){
         int temp=num;
         num=mod;
         mod=temp;
      }
      int r1=num;
      int r2=mod;
      int t1=0;
      int t2=1;
      int q;
      int r;
      int t;
      while(r2>0){
         q=r1/r2;
         r=r1-q*r2;
         r1=r2;
         r2=r;
         t=t1-q*t2;
         t1=t2;
         t2=t;

         if(r1==1) {
            System.out.println("Inverse of num is:- "+t1);
            break;
         }
      }
   }
}
```

9) Find All Primative Roots :

```
import java.util.*;

public class FindAllPrimitiveRoots {
   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;
        }
    }
    System.out.println("<--Primitive Root Table-->");
    for (int i = 1; i <n ; i++) {
        for(int j=1;j<n;j++){
            System.out.print(matrix[i][j]+" ");
        }
        System.out.println();

    }
    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;
            }
        }
    }
```

```
        System.out.println("Order of each element:- ");
        System.out.println(orderOfElement);
        for(Integer ele:orderOfElement.keySet()){

            if(orderOfElement.get(ele)==phi){
                primitiveRootList.add(ele);
            }
        }
        System.out.println(primitiveRootList);
    }
    private static int countNumberOfPrimitiveRoots(){
        return primitiveRootList.size();
    }
    public static void main(String[] args) {
       Scanner sc=new Scanner(System.in);
        System.out.println("Enter any number:-");
        int n=sc.nextInt();
        System.out.println("Calculated Phin:- "+calculatePhi(n));
        primtiveRootTable(n);
        System.out.println("No. of primitive roots:-"+countNumberOfPrimitiveRoots());
    }
}
```

10) HillCipher :

```
import javax.crypto.Cipher;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
public class HillCipher {
    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<>();
        System.out.println("1 FOR ENCRYPTION:");
        System.out.println("2 FOR DECRYPTION:");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 key1=0;
        String ka="";
        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();
            key = key.replaceAll("\\s+", "");
            result = key.matches("[a-z]+");
            if (result == false)
                System.out.println("ENTER CORRECT KEY::");
            else
                f = 1;
        }
            String ans=HillCipher(pt, key);
            System.out.println("Plain Text : "+pt);
            System.out.println("Key: "+key);
            System.out.println("Cipher Text: "+ans);
            break;
    }
}
static String HillCipher(String message, String key) {
    int row=key.length()/3;
    if(key.length()%3!=0){
        row=row+1;
    }
    System.out.println("row:"+row);
    int [][]keyMatrix = new int[row][3];
    getKeyMatrix(key, keyMatrix,row);System.out.println("Key:==>");
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < 3; j++) {
            System.out.print(keyMatrix[i][j]+" ");
        }
        System.out.println();
    }
    int Prow=message.length()/3;
    if(message.length()%3!=0){
        Prow=Prow+1;
    }
    int [][]messageVector = new int[3][Prow];
    getMessageMatrix(message, messageVector,Prow);
    System.out.println("Message:==>");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < Prow; j++) {
            System.out.print(messageVector[i][j]+" ");
        }
        System.out.println();
```

```
        }
        int [][]cipherMatrix = new int[row][Prow];
        encrypt(cipherMatrix, keyMatrix, messageVector,row,Prow);
        System.out.println("Cipher:==>");
        for (int i = 0; i < row; i++) {for (int j = 0; j < Prow; j++) {
           System.out.print(cipherMatrix[i][j]+" ");
        }
           System.out.println();
        }
//
        String CipherText="";
        for (int i = 0; i < row; i++){
           for (int j = 0; j <Prow; j++) {
              CipherText += alphabet1.charAt(cipherMatrix[i][j]);
           }
        }
        return CipherText;
//
    }
    static void getKeyMatrix(String key, int keyMatrix[][],int row)
    {
        int k = 0;
        for (int i = 0; i < row; i++)
        {
           for (int j = 0; j < 3; j++)
           {
              if(k>=key.length()){
                 keyMatrix[i][j]=25;
              }
              else {
                 keyMatrix[i][j] = alphabet.indexOf(key.charAt(k));k++;
              }
           }
        }
    }
    static void getMessageMatrix(String key, int messageVector[][],int row)
    {
        int k = 0;
        for (int i = 0; i < 3; i++)
        {
           for (int j = 0; j < row; j++)
           {
              if(k>=key.length()){
                 messageVector[i][j]=25;
              }
              else {
                 messageVector[i][j] = alphabet.indexOf(key.charAt(k));
                 k++;
              }
           }
        }
    }
    static void encrypt(int cipherMatrix[][], int keyMatrix[][], int messageVector[][],int row,
                    int Prow)
    {
```

```
        int x, i, j;
        for (i = 0; i < row; i++)
        {for (j = 0; j < Prow; j++)
        {
           cipherMatrix[i][j] = 0;
           for (x = 0; x < 3; x++)
           {
              cipherMatrix[i][j] +=
                    keyMatrix[i][x] * messageVector[x][j];
           }
           cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
        }
        }
     }
}
```

11) MillerRabinPrimalityTest :

```
import java.util.Scanner;

public class MillerRabinPrimalityTest {
   private static void rabin(int n,int a)
   {
      int val=n-1;
      int k=0;
      while(val%2==0){
         k++;
         val=val/2;
      }
      System.out.println("Value of m:- "+val);
      System.out.println("Value of k:-"+k);
      int b= (int) (Math.pow(a,val)%n);
      boolean flag=false;
      System.out.println("Initial Value of b after calculation:-"+b);
      if(b==1) System.out.println("n is prime");
      else{
         for(int i=0;i<k;i++) {
            System.out.print("Iteration "+i+":- ");
            System.out.print(b+"*"+b +" mod "+n+" = "+b*b%n);
            System.out.println();
            b=b*b%n;
            if(b==1){
               flag=true;
               break;
            }
            if(b-n==-1){
               System.out.println(b-n +" mod "+n+" = "+-1);
               System.out.println("Prime");
               break;
            }
         }
         if(flag){
            System.out.println("Composite");
```

```
            }
        }
        if(flag==false){
            System.out.println("Composite");
        }
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number:-");
        int n=sc.nextInt();
        System.out.println("Enter the base value:-");
        int a=sc.nextInt();
        rabin(n,a);
    }
}
```

12) MultiplicativeCipher :

```
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);
        }
    }
  }
}
```

13) PlayFairCipher :

```
import java.util.ArrayList;
import java.util.Scanner;
public class PlayFairCipher {
   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");
      System.out.println("1 FOR ENCRYPTION:");
      System.out.println("2 FOR DECRYPTION:");
      System.out.println("Other Key FOR EXIT:");
      int c=sc.nextInt();
      boolean result=false;
      String pt="";String ct="";
      String pt1="";
      String ct1="";
      ArrayList<Integer> spaces=new ArrayList<>();
      int f=0;
      String key="";
      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();
               key=key.replaceAll("\\s+","");
               result = key.matches("[a-z]+");
               if (result == false)
                  System.out.println("ENTER CORRECT KEY::");
               else
                  f = 1;
            }

            String res=encryptByPlayfairCipher(pt, key);
//System.out.println("Plain Text :"+pt1);
            StringBuffer str1 = new StringBuffer(res);
            for (int i = 0; i < pt1.length(); i++) {//
               for (int j = 0; j < spaces.size(); j++) {
                  if (spaces.get(j) == i) {
                     str1.insert(i, ' ');
                  }
               }
            }
            String dec1="";
            dec1 = str1.toString();
            System.out.println("Cipher Text:"+res);
            break;
         case 2:
            while(!result){
               System.out.println("ENTER Cipher Text : ");
               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();
               key=key.replaceAll("\\s+","");
               result = key.matches("[a-z]+");
```

```
                if (result == false)
                    System.out.println("ENTER CORRECT KEY::");
                else
                    f = 1;}
            String res1=dencryptByPlayfairCipher(pt, key);
            System.out.println("Cipher Text :"+pt1);
            StringBuffer str2 = new StringBuffer(res1);
            for (int i = 0; i < pt1.length(); i++) {
                for (int j = 0; j < spaces.size(); j++) {
                    if (spaces.get(j) == i) {
                        str2.insert(i, ' ');
                    }
                }
            }
            String dec2="";
            dec2 = str2.toString();
            System.out.println("Plain Text :"+res1);
            break;

        }
    }
    static String encryptByPlayfairCipher(String pt, String key)
    {
        char arr[][]=new char[5][5];
        String ps="";
        int lengthPT=pt.length();
        if(pt.length()%2==1)
            pt=pt+'z';
        StringBuffer str11 = new StringBuffer(pt);
        int cc=0;
        for (int i = 0; i <lengthPT; i=i+2) {
            if(str11.charAt(i)==str11.charAt(i+1)){
                str11.insert(i+1,'z');
                cc+=1;}
            if(cc==2){
                cc=0;
                lengthPT+=2;
            }
        }
        ps = str11.toString();
        int sizePS=ps.length();
        if(sizePS%2!=0)
            ps+='z';
        System.out.println("Plain Text :"+ps);
        char psA[]=ps.toCharArray();
        int count=0;
// System.out.println(ps);
        char ptA[]=pt.toCharArray();
        char keyA[]=key.toCharArray();
        int sizeK=key.length();
        generateKeyTable(keyA, sizeK, arr);
        System.out.println("KEY FILLED 2D MATRIX:");
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print(arr[i][j]+" ");
```

```
            }
            System.out.println();
        }
        sizePS=ps.length();
        System.out.println("Plain Text :"+ new String(psA));
        String ans=encrypt(psA,arr, sizePS);
        System.out.println("Cipher Text:"+ans);
        return ans;
    }
    static void generateKeyTable(char key[], int ks, char keyT[][])
    {
        int i, j, k, flag = 0;
// a 26 character hashmap
// to store count of the alphabet
        int dicty[]= new int[26];
        for (i = 0; i < ks; i++) {
            if (key[i] != 'j')
                dicty[key[i] - 97] = 2;
        }
        dicty['j' - 97] = 1;
        i = 0;
        j = 0;
        for (k = 0; k < ks; k++) {
            if (dicty[key[k] - 97] == 2) {
                dicty[key[k] - 97] -= 1;
                keyT[i][j] = key[k];
                j++;
                if (j == 5) {
                    i++;
                    j = 0;
                }
            }
        }
        for (k = 0; k < 26; k++) {
            if (dicty[k] == 0) {
                keyT[i][j] = (char)(k + 97);
                j++;if (j == 5) {
                    i++;
                    j = 0;
                }
            }
        }
    }
    static int mod5(int a) { return (a % 5); }
    static String encrypt(char str[], char keyT[][], int ps)
    {
        if(ps%2!=0)
            ps=ps+1;
        int i;
        int a[]=new int[4];
        for (i = 0; i < ps; i += 2) {
            search(keyT, str[i], str[i + 1], a);
            if (a[0] == a[2]) {
                str[i] = keyT[a[0]][mod5(a[1] + 1)];
                str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
```

```
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] + 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
    return new String(str);
}
static void search(char keyT[][], char a, char b, int arr[]){
    int i, j;
    if (a == 'j')
        a = 'i';
    else if (b == 'j')
        b = 'i';
    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            if (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            }
            else if (keyT[i][j] == b) {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}
static String dencryptByPlayfairCipher(String pt, String key)
{
    char arr[][]=new char[5][5];
    String ps=pt;
    int lengthPT=pt.length();
    if(pt.length()%2==1)
        pt=pt+'z';
    StringBuffer str11 = new StringBuffer(pt);
    int cc=0;
    for (int i = 0; i <lengthPT; i=i+2) {
        if(str11.charAt(i)==str11.charAt(i+1)){
            str11.insert(i+1,'z');cc+=1;
        }
        if(cc==2){
            cc=0;
            lengthPT+=2;
        }
    }
    ps = str11.toString();
    int sizePS=ps.length();
    if(sizePS%2!=0)
        ps+='z';
    System.out.println("Cipher Text :"+ps);
    char psA[]=ps.toCharArray();
```

```
        int count=0;
//
        System.out.println(ps);
        char ptA[]=pt.toCharArray();
        char keyA[]=key.toCharArray();
        int sizeK=key.length();
        generateKeyTable(keyA, sizeK, arr);
        System.out.println("KEY FILLED 2D MATRIX:");
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print(arr[i][j]+" ");
            }
            System.out.println();
        }
        sizePS=ps.length();
        System.out.println("Plain Text :"+ new String(psA));
        String ans=decrypt(psA,arr, sizePS);
        System.out.println("Cipher Text:"+ans);
        return ans;}
    static String decrypt(char str[], char keyT[][], int ps)
    {
        if(ps%2!=0)
            ps=ps+1;
        int i;
        int a[]=new int[4];
        for (i = 0; i < ps; i += 2) {
            search(keyT, str[i], str[i + 1], a);
            if (a[0] == a[2]) {
                str[i] = keyT[a[0]][mod5(a[1] - 1)];
                str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
            }
            else if (a[1] == a[3]) {
                int a1=mod5(a[0] - 1);
                if(a1<0)
                    a1=a1+5;
                int a2=mod5(a[2] - 1);
                if(a2<0)
                    a2=a2+5;
                str[i] = keyT[a1][a[1]];
                str[i + 1] = keyT[a2][a[1]];
            }
            else {
                str[i] = keyT[a[0]][a[3]];
                str[i + 1] = keyT[a[2]][a[1]];
            }
        }
        return new String(str);
    }
}
```

14) RSA :

```
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;
        }
    }
}
```

15) Rail Fence :

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

public class RailFence {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in).useDelimiter("\n");
        ArrayList<Integer> spaces=new ArrayList<>();
        boolean result=false;
```

```
    String pt="";
    String pt1="";
    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);
    }
    System.out.println("Enter Depth:");
    int k=sc.nextInt();
    int len=pt.length();
    String ans=encryptRailFence(pt, k);
    System.out.println();
    System.out.println("Encrypted Plain Text:- "+ans);
    System.out.println();
    System.out.println("Original Plain Text:- "+decryptRailFence(ans,k));
}


private static String encryptRailFence(String text, int key){
    char[][] rail=new char[key][(text.length())];
    for (int i=0; i < key; i++)
        for (int j = 0; j < text.length(); j++)
            rail[i][j] ='\f';
    boolean dir_down = false;
    int row = 0, col = 0;


    for (int i=0; i < text.length(); i++)
    {
        if (row == 0 || row == key-1)
            if(dir_down ==true)
                dir_down=false;
            else
                dir_down=true;
        rail[row][col++] = text.charAt(i);
        if(dir_down ==true)
            row+=1;
        else
            row-=1;
    }

    String result = "";
    for (int i=0; i < key; i++)
        for (int j=0; j < text.length(); j++)
            if (rail[i][j]!='\f')
                result += rail[i][j];

    for (int i=0;i<key;i++){
```

```
        for (int j=0; j < text.length(); j++){
            System.out.print(rail[i][j]+" ");
        }
        System.out.println();
    }

    return result;
}



private static String decryptRailFence(String text, int key){
    char[][] rail=new char[key][(text.length())];
    for (int i=0; i < key; i++)
        for (int j = 0; j < text.length(); j++)
            rail[i][j] ='\f';
    boolean dir_down = false;

    int row = 0, col = 0;

    for (int i=0; i < text.length(); i++)
    {
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;
        rail[row][col++] = '*';

        if(dir_down ==true)
            row+=1;
        else
            row-=1;
    }
    int index = 0;
    for (int i=0; i<key; i++)
        for (int j=0; j<text.length(); j++)
            if (rail[i][j] == '*' && index<text.length())
                rail[i][j] = text.charAt(index++);
    String result="";

    row = 0;
    col = 0;
    for (int i=0; i< text.length(); i++)
    {
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;


        if (rail[row][col] != '*')
            result += rail[row][col++];

        if(dir_down ==true)
            row+=1;
```

```
            else
                row-=1;
        }
        return result;
    }
}
```