Certificate

This is to Certify that

Mr./Miss Shaikh Farhaan Mujib

Studying in BECSE II

has performed set of experiments

of the subject CNS

Satisfactorily in the year 2019-20

Date :

Head of Dept. Incharge

INDEX

INDEX

Experiment - 01

Aim → Write a Java program to implement general caesar cipher.

Theory →

↳ The caesar cipher is one of the earliest and simplest method of encryption technique.

↳ It is simply a type of substitution cipher i.e. each letter of alphabet is replaced by another letter.

↳ For eg. Consider the key or shift is 3 and A is replaced by D then B will be replaced by E.

↳ It is simple & traditional technique, but it is easy to decode.

↳ Encryption and Decrypt^{n} in caesar cipher can be represented as;

1. Encryption;

$$E_n(x) = (x+n) \bmod 26$$

Where,
n is shift / key

2. Decryption;

$$D_n(x) = (x-n) \bmod 26$$

Ex. plaintext = "fuki" & shift is 2;

F U K I
↓ ↓ ↓ ↓
H W M K

ciphertext = "HWMK"

Conclusion →

Thus we have implemented a Java program for encryption / decryption using caesar cipher.

NAME: FURKHAN MUJIBODDEN SHAIKH

CLASS: BECSEII

ROLL:63

AIM: WRITE A JAVA PROGRAM TO IMPLEMENT CAESAR CIPHER

```
import java.util.*;
class CaesarCipher {

   Scanner sc = new Scanner(System.in);
   char alphabets[] = new char[26];
   int i;
   char j;
   String input = new String();
   int key;
   char cipherText[] = new char[200];
   char plainText[] = new char[200];

   String strcipherText="";

   CaesarCipher(){

      for(i=0,j='A';i<26;i++,j++){
         alphabets[i] = j;
      }

   }

   void getInputs(){
      System.out.println("\nEnter the Input String To Be Encrypted");
      input = sc.nextLine();
      input = input.toUpperCase();
      // System.out.println(input);
   }

   void getKey(){
      System.out.println("\nEnter the Key");
      key = sc.nextInt();
      // System.out.print(key);
   }

   int getIndex(char ch){
      int temp = (int)ch;
      int tmp=0;
      int temp_integer = 64;
      if(temp<=90 & temp>=65)
      {
         tmp = temp - temp_integer - 1;
         return tmp;
      }
      return 0;

   }

   void cipher(){
      int max = 25;
      int i;
      int index;
      int replacement_index;
      for(i=0;i<input.length();i++){
         if(Character.isLetter(input.charAt(i))){
            index = getIndex(input.charAt(i));
            if( (index+key) > max){
               replacement_index = (index+key) - max - 1;
               cipherText[i] = alphabets[replacement_index];
            }
            else{
               cipherText[i] = alphabets[index + key];
            }
         }
         else{
```

```
        cipherText[i] = input.charAt(i);
      }
    }

    System.out.print(input + " is Encrypted to ");
    for(i=0;i<cipherText.length;i++){
      System.out.print(cipherText[i]);
    }
    System.out.println();

}

void decrypt(){

    int min = 0;
    int i;
    int replacement_index;
    int index;

    for(i=0;i<cipherText.length;i++){
      strcipherText += cipherText[i];
    }
    for(i=0;i<strcipherText.length();i++){
      if(Character.isLetter(strcipherText.charAt(i))){
        index = getIndex(strcipherText.charAt(i));
        if( (index-key) < min){
          replacement_index = (index+key)  - 1;
          plainText[i] = alphabets[replacement_index];
        }
        else{
          plainText[i] = alphabets[index - key];
        }
      }
      else{

        plainText[i] = strcipherText.charAt(i);
      }
      }

      System.out.print(strcipherText.trim() + " is Decrypted to
");
      for(i=0;i<plainText.length;i++){
        System.out.print(plainText[i]);
      }
      System.out.println();



    }
}

class caesarFurkhan {
   public static void main(String args[]){
      CaesarCipher cs = new CaesarCipher();
      cs.getInputs();
      cs.getKey();
      cs.cipher();
      cs.decrypt();
   } }
```

+++++OUTPUT+++++

Enter the Input String To Be Encrypted

FURKHAN MUJIBODDEN SHAIKH

Enter the Key

4

FURKHAN MUJIBODDEN SHAIKH is Encrypted to JYVOLER QYNMFSHHIR WLEMOL

JYVOLER QYNMFSHHIR WLEMOL is Decrypted to FURKHAN MUJIBODDEN SHAIKH

Experiment - 02

Aim → Write a program to implement monoalphabetic cipher.

Theory →

↳ With only 25 possible keys, the caesar cipher is far from secure.

↳ If, instead, the "cipher" line can be any permutation of 26 alphabetic characters, then there are 26! or more than 4×10^{26} possible keys.

↳ This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate cryptanalysis.

↳ There is however another line of attack. If the hacker knows the nature of plaintext, then it easy to crack

↳ the relative frequency of the letters can be determined and compared to a standard frequency distribution of english.

Ex. plaintext = "Hello world"
key = "GET"

Table;

plain alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
cipher alphabet	G	E	T	A	B	C	D	F	H	I	J	K	L	M	N	O	P	Q	R	S	U	V	W	X	Y	Z

∴ ciphertext = DBKK WNQKA

↳ security of this cipher is good as compared to caesar cipher but it can be easily decoded.

Conclusion → Thus we have implemented a Java program for monoalphabic cipher.

NAME: FURKHAN MUJIBODDEN SHAIKH

CLASS: BECSEII

ROLL:63

AIM: WRITE A JAVA PROGRAM TO IMPLEMENT MONOALPHABETIC CIPHER

```
import java.util.*;
class monoCipher {

  Scanner sc = new Scanner(System.in);
  char alphabets[] = new char[26];
  char cipherbets[] = new char[26];
  String input = new String();
  String key;
  String ciphertext = new String();
  String plaintext = new String();

  monoCipher(){
    int i;
    char j;
    for(i=0,j='A';i<26;i++,j++){
      alphabets[i] = j;
    }
  }

  boolean check(char ch){
    for(int i=0;i<cipherbets.length;i++){
      if(ch == cipherbets[i]){
        return true;
      }
    }
    return false;
  }

  void setCipherAlphabets(){
    int cnt=0;
    for(int i=0;i<key.length();i++){
      cipherbets[i] = key.charAt(i);
```

```
    }
    for(int j=0;j<26;j++){
        if(check(alphabets[j])){
            continue;
        }
        else{
            if((key.length()+cnt) < 26)
            {
                cipherbets[key.length()+cnt] = alphabets[j];
                cnt+=1;
            }
        }
    }

}



void getInputs(){
    System.out.println("\nEnter the Input String To Be Encrypted");
    input = sc.nextLine();
    input = input.toUpperCase();
    // System.out.println(input);
}

void getKey(){
    System.out.println("\nEnter the Key");
    key = sc.nextLine();
    key = key.toUpperCase();
    // System.out.print(key);
}

int getIndex(char ch){
    int temp = (int)ch;
    int tmp=0;
    int temp_integer = 64;
    if(temp<=90 & temp>=65)
    {
```

```
        tmp = temp - temp_integer - 1;
        return tmp;
    }
    return 0;

}

int getIndexInCipherBets(char ch){
    for(int i=0;i<cipherbets.length;i++){
        if(ch == cipherbets[i]){
            return i;
        }
    }
    return -1;
}

void cipher(){
    System.out.print("\nOpen Alphabet   : ");
    for(int j=0;j<26;j++){
        System.out.print(alphabets[j] +" ");
    }
    System.out.println();

    System.out.print("Cipher Alphabet : ");
    for(int j=0;j<26;j++){
        System.out.print(cipherbets[j] +" ");
    }
    System.out.println();

    int index;
    ciphertext = "";

    for(int i=0;i<input.length();i++){
        if(Character.isLetter(input.charAt(i))){
            index = getIndex(input.charAt(i));
            ciphertext+=Character.toString(cipherbets[index]);
        }
```

```
        else{
            ciphertext += input.charAt(i);

        }

    }
    System.out.println("\n" + input +" is Encrypted to " +ciphertext);
}

void decrypt(){

    System.out.print("\nCipher Alphabet : ");
    for(int j=0;j<26;j++){
        System.out.print(cipherbets[j] +" ");
    }
    System.out.println();

    System.out.print("Open Alphabet   : ");
    for(int j=0;j<26;j++){
        System.out.print(alphabets[j] +" ");
    }
    System.out.println();

    int index;
    for(int i=0;i<ciphertext.length();i++){
        if(Character.isLetter(ciphertext.charAt(i))){
            index = getIndexInCipherBets(ciphertext.charAt(i));
            plaintext+=Character.toString(alphabets[index]);
        }
        else{
            plaintext += input.charAt(i);

        }

    }
    System.out.println("\n" + ciphertext +" is Decrypted to " +plaintext);
}
```

```
}

class monoAlphabeticCipher {
   public static void main(String args[]){
      monoCipher mc = new monoCipher();
      mc.getInputs();
      mc.getKey();
      mc.setCipherAlphabets();
      mc.cipher();
      mc.decrypt();
   }

}
```

+++++OUTPUT+++++

Enter the Input String To Be Encrypted

FURKHAN SHAIKH

Enter the Key

SHAIK

Open Alphabet : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Cipher Alphabet : S H A I K B C D E F G J L M N O P Q R T U V W X Y Z

FURKHAN SHAIKH is Encrypted to BUQGDSM RDSEGD

Ciher Alphabet : S H A I K B C D E F G J L M N O P Q R T U V W X Y Z

Opn Alphabet : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

BUQGDSM RDSEGD is Decrypted to FURKHAN SHAIKH

Experiment → 03

Aim → WAP to implement poly-alphabetic cipher.

Theory →

↳ Vigenere cipher is a method of encrypting alphabetic text.

↳ It uses a simple form of polyalphabetic cipher.

↳ A Vigenere cipher is a cipher based on multiple substitution.

↳ The encryption of plaintext is done by Vigenere table. This table consist of alphabets written out 26 times rows, each alphabet shifted cyclically to left.

↳ At different points in encryption process, it uses alphabet from row.

↳ At different points in encrypt process, the cipher uses a different alphabet from one of the rows.

↳ The alphabet at each point depends on repeating a keyword.

Input: plaintext: "GEEKS"
key: "Ayush"

ciphertext: GCYCZ

The vigenere table is generated and By using the key "Ayush" we have generated a ciphertext "GCYCZ"

Conclusion →

Thus we have implemented the polyalphabetic cipher in Java.

NAME: FURKHAN MUJIBODDEN SHAIKH

CLASS: BECSEII

ROLL:63

AIM: WRITE A JAVA PROGRAM TO IMPLEMENT POLYALPHABETIC CIPHER

```
import java.util.*;
class monoCipher {

  Scanner sc = new Scanner(System.in);
  char alphabets[] = new char[26];
  char cipherbets[] = new char[26];
  String input = new String();
  String key;
  String ciphertext = new String();
  String plaintext = new String();

  monoCipher(){
    int i;
    char j;
    for(i=0,j='A';i<26;i++,j++){
      alphabets[i] = j;
    }
  }

  boolean check(char ch){
    for(int i=0;i<cipherbets.length;i++){
      if(ch == cipherbets[i]){
        return true;
      }
    }
    return false;
  }

  void setCipherAlphabets(){
    int cnt=0;
    for(int i=0;i<key.length();i++){
      cipherbets[i] = key.charAt(i);
    }
    for(int j=0;j<26;j++){
      if(check(alphabets[j])){
        continue;
      }
      else{
          if((key.length()+cnt) < 26)
          {
            cipherbets[key.length()+cnt] = alphabets[j];
            cnt+=1;
          }
      }
    }

  }


  void getInputs(){
    System.out.println("\nEnter the Input String To Be Encrypted");
    input = sc.nextLine();
    input = input.toUpperCase();
    // System.out.println(input);
  }

  void getKey(){
    System.out.println("\nEnter the Key");
    key = sc.nextLine();
    key = key.toUpperCase();
    // System.out.print(key);
  }

  int getIndex(char ch){
    int temp = (int)ch;
    int tmp=0;
    int temp_integer = 64;
    if(temp<=90 & temp>=65)
    {
```

```
        tmp = temp - temp_integer - 1;
        return tmp;
    }
    return 0;

}

int getIndexInCipherBets(char ch){
    for(int i=0;i<cipherbets.length;i++){
        if(ch == cipherbets[i]){
            return i;
        }
    }
    return -1;
}

void cipher(){
    System.out.print("\nOpen Alphabet   : ");
    for(int j=0;j<26;j++){
        System.out.print(alphabets[j] +" ");
    }
    System.out.println();

    System.out.print("Cipher Alphabet : ");
    for(int j=0;j<26;j++){
        System.out.print(cipherbets[j] +" ");
    }
    System.out.println();

    int index;
    ciphertext = "";

    for(int i=0;i<input.length();i++){
        if(Character.isLetter(input.charAt(i))){
            index = getIndex(input.charAt(i));
            ciphertext+=Character.toString(cipherbets[index]);
        }
        else{
            ciphertext += input.charAt(i);

        }

    }
    System.out.println("\n" + input +" is Encrypted to "
+ciphertext);
  }

  void decrypt(){

    System.out.print("\nCipher Alphabet : ");
    for(int j=0;j<26;j++){
        System.out.print(cipherbets[j] +" ");
    }
    System.out.println();

    System.out.print("Open Alphabet   : ");
    for(int j=0;j<26;j++){
        System.out.print(alphabets[j] +" ");
    }
    System.out.println();

    int index;
    for(int i=0;i<ciphertext.length();i++){
        if(Character.isLetter(ciphertext.charAt(i))){
            index = getIndexInCipherBets(ciphertext.charAt(i));
            plaintext+=Character.toString(alphabets[index]);
        }
        else{
            plaintext += input.charAt(i);

        }

    }
    System.out.println("\n" + ciphertext +" is Decrypted to "
+plaintext);
```

```
  }


}


class monoAlphabeticCipher {
   public static void main(String args[]){
      monoCipher mc = new monoCipher();
      mc.getInputs();
      mc.getKey();
      mc.setCipherAlphabets();
      mc.cipher();
      mc.decrypt();
   }


}
```

+++++OUTPUT+++++

Enter the Input String To Be Encrypted

Furkhan MUJIBODDEN SHAIKH

Enter the Key

SHAIKH

Extended Key is SHAIKHS HAIKHSHAIK HSHAIK

FURKHAN MUJIBODDEN SHAIKH is Encerypted To

XBRSRHF TURSIGKDMX ZZHISR

XBRSRHF TURSIGKDMX ZZHISR is DECRYPTED To

FURKHAN MUJIBODDEN SHAIKH

Experiment → 04

Aim → WAP to find multiplicative inverse using Euclidean Algorithm.

Theory →

↳ Given two integers named 'a' and 'm', so find out modular arithmetic;

$$ax \equiv 1 \pmod{m}$$

The value of x should be in $\{0, 1, 2, \ldots m-1\}$, it means x should be in the range of integer

↳ The multiplicative inverse of "a modulo m" exists iff a & m are relatively prime.

Ex. $GCD(a, m) = 1$

Ex.

a = 3 , m = 11

since (4 * 3) mod 11 = 1, 4 is modulo inverse of 3 (under 11).

so output will be "4"

a = 10 , m = 17

since (10 * 12) mod 17 = 1 then 12 is modulo inverse of 10.

Conclusion →

Thus we have written a Java program to find multiplicative inverse using Euclidean Algorithm.

NAME: FURKHAN MUJIBODDEN SHAIKH

CLASS: BECSEII

ROLL:63

AIM: WRITE A JAVA PROGRAM TO CALCULATE MULTIPLICATIVE INVERSE

```
import java.util.*;
class monoCipher {

  Scanner sc = new Scanner(System.in);
  char alphabets[] = new char[26];
  char cipherbets[] = new char[26];
  String input = new String();
  String key;
  String ciphertext = new String();
  String plaintext = new String();

  monoCipher(){
    int i;
    char j;
    for(i=0,j='A';i<26;i++,j++){
      alphabets[i] = j;
    }
  }

  boolean check(char ch){
    for(int i=0;i<cipherbets.length;i++){
      if(ch == cipherbets[i]){
        return true;
      }
    }
    return false;
  }

  void setCipherAlphabets(){
    int cnt=0;
    for(int i=0;i<key.length();i++){
      cipherbets[i] = key.charAt(i);
    }
    for(int j=0;j<26;j++){
      if(check(alphabets[j])){
        continue;
      }
      else{
        if((key.length()+cnt) < 26)
        {
          cipherbets[key.length()+cnt] = alphabets[j];
          cnt+=1;
        }
      }
    }

  }


  void getInputs(){
    System.out.println("\nEnter the Input String To Be Encrypted");
    input = sc.nextLine();
    input = input.toUpperCase();
    // System.out.println(input);
  }

  void getKey(){
    System.out.println("\nEnter the Key");
    key = sc.nextLine();
    key = key.toUpperCase();
    // System.out.print(key);
  }

  int getIndex(char ch){
    int temp = (int)ch;
    int tmp=0;
    int temp_integer = 64;
```

```
    if(temp<=90 & temp>=65)
    {
        tmp = temp - temp_integer - 1;
        return tmp;
    }
    return 0;

}

int getIndexInCipherBets(char ch){
    for(int i=0;i<cipherbets.length;i++){
        if(ch == cipherbets[i]){
            return i;
        }
    }
    return -1;
}

void cipher(){
    System.out.print("\nOpen Alphabet   : ");
    for(int j=0;j<26;j++){
        System.out.print(alphabets[j] +" ");
    }
    System.out.println();

    System.out.print("Cipher Alphabet : ");
    for(int j=0;j<26;j++){
        System.out.print(cipherbets[j] +" ");
    }
    System.out.println();

    int index;
    ciphertext = "";

    for(int i=0;i<input.length();i++){
        if(Character.isLetter(input.charAt(i))){
            index = getIndex(input.charAt(i));
            ciphertext+=Character.toString(cipherbets[index]);
        }
        else{
            ciphertext += input.charAt(i);

        }

    }
    System.out.println("\n" + input +" is Encrypted to "
+ciphertext);
    }

    void decrypt(){

        System.out.print("\nCipher Alphabet : ");
        for(int j=0;j<26;j++){
            System.out.print(cipherbets[j] +" ");
        }
        System.out.println();

        System.out.print("Open Alphabet   : ");
        for(int j=0;j<26;j++){
            System.out.print(alphabets[j] +" ");
        }
        System.out.println();

        int index;
        for(int i=0;i<ciphertext.length();i++){
            if(Character.isLetter(ciphertext.charAt(i))){
                index = getIndexInCipherBets(ciphertext.charAt(i));
                plaintext+=Character.toString(alphabets[index]);
            }
            else{
                plaintext += input.charAt(i);

            }

        }
```

```
    System.out.println("\n" + ciphertext +" is Decrypted to "
+plaintext);
  }


}


class monoAlphabeticCipher {
  public static void main(String args[]){
    monoCipher mc = new monoCipher();
    mc.getInputs();
    mc.getKey();
    mc.setCipherAlphabets();
    mc.cipher();
    mc.decrypt();
  }
```

```
}
```

+++++OUTPUT+++++

Program Calculates Multiplicative Inverse of M % N

Enter the Value of M

5678

Enter the Value of M

8765

Multiplicative Inverse of 5678 % 8765 is 2527

Experiment - 05

Aim → Write a program to implement hill cipher.

Theory →

↳ Hill cipher is a polygraphic substitution cipher, based on linear algebra.

↳ Each letter is represented by a number modulo 26.

↳ To encrypt a message, each block of n letters is multiplied by an invertible n×n matrix.

↳ The matrix used for encryption is the cipher key, and it should be choosen used for encryption.

↳ The matrix used for encryption is invertible.

Ex.

Input: ACT

Key: GYBNQKURP

Ciphertext: POH

We have to encrypt the message 'ACT' n = 3.

So key matrix is;

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} = M$$

The message ACT is written as vector;

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = N, \quad \therefore MXN = \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix}$$

∴ Ciphertext is POH.

Conclusion →

Thus we have implemented a Java program for hill cipher.

NAME: FURKHAN MUJIBODDEN SHAIKH

CLASS: BECSEII

ROLL:63

AIM: WRITE A JAVA PROGRAM TO IMPLEMENT HILL CIPHER

```
import java.util.*;

class HillCipher {

   Scanner sc = new Scanner(System.in);
   char alphabets[] = new char[26];
   String input = new String();
   int [][]key;
   String ciphertext = new String();
   String plaintext = new String();
   int vector[][];
   int len=0;
   int rows,cols;
   String cipherstring="";

   HillCipher(){
      int i;
      char j;
      for(i=0,j='A';i<26;i++,j++){
         alphabets[i] = j;
      }
   }



   void getInputs(){
      System.out.println("\nEnter the Input String To Be
Encrypted");
      input = sc.nextLine();
      input = input.toUpperCase();
   }


   String multiply(int a[][],int b[][]){
      int c[][] = new int[rows][1];
      for(int i=0;i<rows;i++){
         for(int j=0;j<1;j++){
            c[i][j]=0;
            for(int k=0;k<rows;k++)
            {
               c[i][j]+=a[i][k]*b[k][j];
            }
         }
      }

      int x;
      String res="";
      for(int i=0;i<rows;i++){
         for(int j=0;j<1;j++){
            x = c[i][j];
            x = x % 26;
            res+= String.valueOf(x);
         }
         res+=" ";
      }
      return res.trim();
   }

   void setVectors(){
      len= input.length();
      int j=0;

      if(len%2 == 1){
         input+="X";
      }
      for(int i=0;i<(input.length())/2;i++){
         for(int k=0;k<rows;k++){
            vector[k][0] = getIndex(input.charAt(j));
            j+=1;
         }
         cipherstring += multiply(key,vector);
```

```
      cipherstring+=" ";
    }
    String cipherarr[] = cipherstring.split("\\s+");
    // for(int i=0;i<cipherarr.length;i++){
    //    System.out.print(cipherarr[i] +" ");
    // }
    for(int i=0;i<cipherarr.length;i++){
      ciphertext+=getChar(Integer.parseInt(cipherarr[i]));
    }
    System.out.println("\n" + input+ " is Encrypted to "
+ciphertext);
    System.out.println("\n" + ciphertext +" is Decrypted to "
+input);

  }

  void getKey(){
    System.out.println("\nEnter Rows in Key Matrix");
    rows = sc.nextInt();
    System.out.println("Enter Columns in Key Matrix");
    cols = sc.nextInt();

    key = new int[rows][cols];
    vector = new int[rows][1];

    System.out.println("Enter the Matrix Contents Row-
Wise");
    for(int i=0;i<rows;i++){
      for(int j=0;j<cols;j++){
        key[i][j] = sc.nextInt();
      }
    }

    System.out.print("\nKey is : \n");
    for(int i=0;i<rows;i++){
      for(int j=0;j<cols;j++){
        System.out.print(key[i][j] + " ");
      }
        System.out.println();
      }

      setVectors();

  }

  int getIndex(char ch){
    int temp = (int)ch;
    int tmp=0;
    int temp_integer = 64;
    if(temp<=90 & temp>=65)
    {
      tmp = temp - temp_integer - 1;
      return tmp;
    }
    return 0;
  }

  char getChar(int ch){
    return alphabets[ch];
  }

}

class HillCipherFurkhan {
  public static void main(String args[]){
    HillCipher hc = new HillCipher();
    hc.getInputs();
    hc.getKey();
  }

}
```

+++++OUTPUT+++++

Enter the Input String To Be Encrypted

goodmorning

Enter Rows in Key Matrix

2

Enter Columns in Key Matrix

2

Enter the Matrix Contents Row-Wise

3 5

2 7

Key is :

3 5

2 7

GOODMORNINGX is Encrypted to KGFXCSMVLDDR

KGFXCSMVLDDR is Decrypted to GOODMORNINGX

Experiment - 06

Aim → Write a program to implement transposition cipher.

Theory →

↳ The columnar transposition is a form of transposition cipher, just like rail fence cipher.

↳ This method involves writing the plaintext out in rows, and then reading the ciphertext off in columns.

↳ In transposition cipher, the order of alphabets is re-arranged to obtain the ciphertext.

↳ Width of the rows and the permutation is defined by a keyword.

↳ Finally, the message is read off in columns, in order to specify by the keyword.

Ex.

Input: Geeks for Geeks.

key: HACK Length: 4 order: 3124

H	A	C	K
3	1	2	4
G	e	e	K
S	f	o	r
G	e	e	K
S			

We will print characters of 1st col order 1, 2, 3, 4

So, 1 → eefe

2 → eoe

3 → GSGS

4 → KrK

∴ ciphertext is efeeoegsgskrk.

Conclusion →

Thus we have written a Java code to implement transposition cipher.

```
/*
Name: Furkhan Mujibodden Shaikh
Class: BECSEII
Aim: WRITE A JAVA PROGRAM TO IMPLEMENT TRANSPOSITION CIPHER
ROLL: 63
*/

import java.util.*;

class TranspositionCipher {

  Scanner sc = new Scanner(System.in);
  char alphabets[] = new char[26];
  String input = new String();
  int no_of_digits_in_key;
  int key[] = new int[20];
  String ciphertext = new String();
  String plaintext = new String();
  int cols,rows;
  char table[][];
  char table2[][];

  TranspositionCipher(){
    int i;
    char j;
    for(i=0,j='A';i<26;i++,j++){
      alphabets[i] = j;
    }
  }

  int max(int a[],int len){
    int m = a[0];
    for(int i=1;i<len;i++){
      if(m<a[i]){
        m = a[i];
      }
      else{
        continue;
      }
    }
    return m;
  }

  void getInputs(){
    int cnt=0;
    System.out.println("\nEnter the Input String To Be Encrypted");
    input = sc.nextLine();
    input = input.toUpperCase();
    input.trim();

    for(int i=0;i<input.length();i++){
      if(input.charAt(i) == 32){
        break;
      }
      cnt+=1;
    }
  }

  void getKey(){
    System.out.println("\nEnter the Space Separated Key");
    String st;
    String tmp[];
    st = sc.nextLine();
    tmp = st.split(" ");
    no_of_digits_in_key = tmp.length;
    for(int i=0;i<no_of_digits_in_key;i++){
      key[i] = Integer.parseInt(tmp[i]);
    }
    cols = max(key,no_of_digits_in_key);
    int temp;
    temp = input.length()%cols;
    if(temp == 0){
      rows = input.length() / cols;
```

```
    }
    else{
        rows = input.length()/cols +1;
    }

    table = new char[rows][cols];
    table2 = new char[rows][cols];

    System.out.println();
}

void cipher(){
    int k=0;
    System.out.println("\nCipher Matrix: ");
    for(int i=1;i<=no_of_digits_in_key;i++){
        System.out.print(i +" ");
    }
    System.out.println("");

    for(int i=0;i<rows;i++){

        for(int j=0;j<cols;j++){
            if(k < input.length()){
                table[i][j] = input.charAt(k);
                k+=1;
            }

        }
    }

    for(int i=0;i<rows;i++){
        for(int j=0;j<cols;j++){
            System.out.print(table[i][j] +" ");
        }
        System.out.println();
    }
    System.out.println();

        int i=key[0];
        int ctr=0;
        // i refer to columns
        // j refer to rows
        for(;ctr<no_of_digits_in_key;){
            for(int j=0;j<rows;j++){
                ciphertext+=table[j][i-1];
            }
            i = key[++ctr];
        }
        ciphertext = ciphertext.replaceAll("\0", "");
        System.out.println(input +" is Encrypted To "
+ciphertext);
    }

    void decrypt(){
        int k=0;
        System.out.println("\nPlain Matrix: ");
        for(int i=1;i<=no_of_digits_in_key;i++){
            System.out.print(i +" ");
        }
        System.out.println("");

        int i = key[0];
        int ctr=0;
        for(;ctr<no_of_digits_in_key;){
            for(int j=0;j<rows;j++){
                if(k<input.length()){
                    table2[j][i-1]= ciphertext.charAt(k);
                    k+=1;
                }
            }
            i = key[++ctr];
        }

        for(i=0;i<rows;i++){
            for(int j=0;j<cols;j++){
```

```
            System.out.print(table2[i][j] +" ");
        }
        System.out.println();
    }
    System.out.println();

    for(i=0;i<rows;i++){
        for(int j=0;j<cols;j++){
            plaintext+=table2[i][j];
        }
    }

    System.out.println(ciphertext +" is Decyrted To "
+plaintext);
  }

}

class TranspositionCipherFurkhan {
  public static void main(String args[]){
    TranspositionCipher tc = new TranspositionCipher();
    tc.getInputs();
    tc.getKey();
    tc.cipher();
    tc.decrypt();
  }
}

// OUTPUT
/*
Enter the Input String To Be Encrypted
FURKHAN SHAIKH

Enter the Space Separated Key
4 3 2 1 5

Cipher Matrix:
1 2 3 4 5
F U R K H
A N  S H
A I K H

FURKHAN SHAIKH is Encrypted To KSHR KUNIFAAHH

Plain Matrix:
1 2 3 4 5
F U R K H
A N  S H
A I K H

KSHR KUNIFAAHH is Decyrted To FURKHAN SHAIKH
*/
```

Experiment - 07

Aim → WAP to implement RSA encryption Algorithm.

Theory →

↳ RSA algorithm is asymmetric cryptography algorithm. As Asymmetric actually means that it works on two different keys.

1. public key.
2. Private key.

↳ public key is shared with Everyone, while private key is kept secret.

↳ Eg.

1. A client sends its public key to the server & request some data.

2. The server encrypts the data using client's public key & sends encrypted data.

3. Client receives the data and decrypts it.

→ Since this is Asymmetric, nobody can decrypt the data except browser even if 3rd party has public key of Browser.

Generating public key;

Let p = 53 and Q = 59

so, p*q = n

→ 1st part of public key

Generating Private key;

$\phi(n)$:

= (p-1)(Q-1)

so, $\phi(n)$ = 3016

Conclusion →

Thus we have implemented RSA encryption using Java.

```
/*
Name: Furkhan Mujibodden Shaikh
Class: BECSEII;
Aim: WAP to implement RSA ENCRYPTION ALGORITHM
*/

import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

import javax.crypto.Cipher;
public class RSAEncryptionFurkhan {

   public static void main(String[] args) throws Exception {

      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the Input String");
      String plainText = sc.nextLine();

      Map<String, Object> keys = getRSAKeys();

      PrivateKey privateKey = (PrivateKey) keys.get("private");
      PublicKey publicKey = (PublicKey) keys.get("public");

      String encryptedText = encryptMessage(plainText,
privateKey);
      String decryptedText = decryptMessage(encryptedText,
publicKey);

      System.out.println();
      System.out.println(plainText +" is Encrypted to [ "
+encryptedText +" ]\n");
      System.out.println("[ " +encryptedText + " ] is Decrypted
to " + decryptedText);
      sc.close();

   }

   // Get RSA keys. Uses key size of 2048.
   private static Map<String,Object> getRSAKeys() throws
Exception {
      KeyPairGenerator keyPairGenerator =
KeyPairGenerator.getInstance("RSA");
      keyPairGenerator.initialize(2048);
      KeyPair keyPair = keyPairGenerator.generateKeyPair();
      PrivateKey privateKey = keyPair.getPrivate();
      PublicKey publicKey = keyPair.getPublic();

      Map<String, Object> keys = new
HashMap<String,Object>();
      keys.put("private", privateKey);
      keys.put("public", publicKey);
      return keys;
   }
  private static String decryptMessage(String encryptedText,
PublicKey publicKey) throws Exception {
      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.DECRYPT_MODE, publicKey);
      return new
String(cipher.doFinal(Base64.getDecoder().decode(encrypte
dText)));
   }
   private static String encryptMessage(String plainText,
PrivateKey privateKey) throws Exception {
      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.ENCRYPT_MODE, privateKey);
      return
Base64.getEncoder().encodeToString(cipher.doFinal(plainTex
t.getBytes()));
   }}

// OUTPUT
/*
Enter the Input String
Furkhan Mujibodden Shaikh
```

Furkhan Mujibodden Shaikh is Encrypted to [GRwaRAkxJimNEvtAV8DDPkENPGomVAPQLZ1qGtnoEUE/3iU jqmb05NT8rqx5d/mFGLRzK2p5IMLk3DcknTHIYo0eRQjdeSx8 n0CKGOVCEEamvLle92ueDWg7MCBTwtpeTKuulv1EKKRmaU aYc150SDFui6jDCSUO/AuHBqj7MGvMUAzNJUNNXJu84YZek QCu0HP7nHmoXpogwNSSoSMMsinfNWt/3i17LzwODnHhRu2 dZE0fy/SqA6zQ6X3MrncVuKdgmMMdqzeklt41JGzYEntFsExG mImrzmH4E6MEz8MMxpdQwHSTggSEKVdgahYUoW26OWvf yJXpWWZ5eoT6sQ==]

[GRwaRAkxJimNEvtAV8DDPkENPGomVAPQLZ1qGtnoEUE/3iU jqmb05NT8rqx5d/mFGLRzK2p5IMLk3DcknTHIYo0eRQjdeSx8 n0CKGOVCEEamvLle92ueDWg7MCBTwtpeTKuulv1EKKRmaU aYc150SDFui6jDCSUO/AuHBqj7MGvMUAzNJUNNXJu84YZek QCu0HP7nHmoXpogwNSSoSMMsinfNWt/3i17LzwODnHhRu2 dZE0fy/SqA6zQ6X3MrncVuKdgmMMdqzeklt41JGzYEntFsExG mImrzmH4E6MEz8MMxpdQwHSTggSEKVdgahYUoW26OWvf yJXpWWZ5eoT6sQ==] is Decrypted to Furkhan Mujibodden Shaikh*/

Experiment → 08

Aim → WAp to implement Diffie-Hellman key exchange algorithm.

Theory →

↳ the Diffie-Hellman algo is being used to establish a shared secret that can be used for secret.

↳ Communications while exchanging data over a public network using elliptic curve to generate points & get the secret key using parameters.

↳ For sake of simplicity & practical implementation of algorithm, we will consider only 4 variables one prime p and G and two private values a and b.

↳ p & G both are relatively prime.

↳ Users say Bob and Alice pick private values of a & b. Now they generate key and exchange it publicly.

ALICE	BOB
public keys available = P, G.	public keys are P, g
private key selected = a	private key selected = b.
Exchange of keys	
key got = y	key got = x
secret key = $K_a = y^a \bmod p$	secret key $K_b = x^b \bmod p$
$K_a = K_b$	
Users now have symmetric key	

Conclusion →

Thus we have written a Java program to implement Diffie-hellman key exchange algorithm.

NAME: FURKHAN MUJIBODDEN SHAIKH

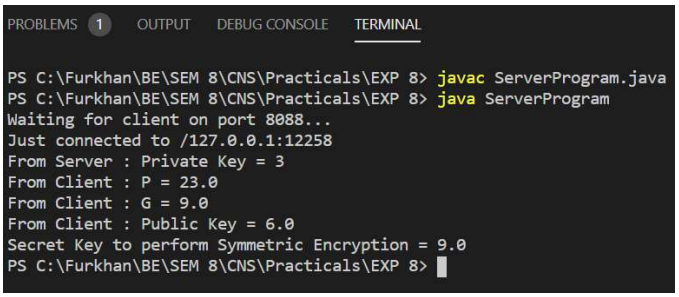
CLASS: BECSEII

AIM: WRITE A PROGRAM TO IMPLEMENT DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

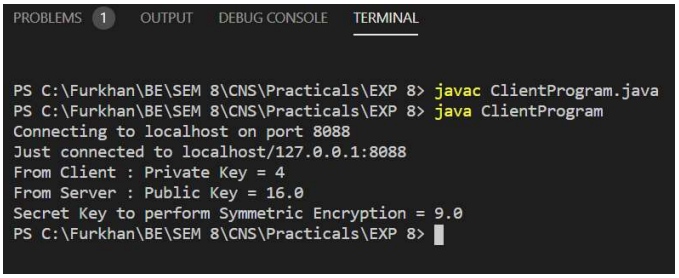
```
import java.net.*;
import java.io.*;

public class ClientProgram {
   public static void main(String[] args)
   {
      try {
         String pstr, gstr, Astr;
         String serverName = "localhost";
         int port = 8088;

         // Declare p, g, and Key of client
         int p = 23;
         int g = 9;
         int a = 4;
         double Adash, serverB;

         // Established the connection
         System.out.println("Connecting to " + serverName
                  + " on port " + port);
         Socket client = new Socket(serverName, port);
         System.out.println("Just connected to "
                  + client.getRemoteSocketAddress());

         // Sends the data to client
         OutputStream outToServer =
client.getOutputStream();
         DataOutputStream out = new
DataOutputStream(outToServer);

         pstr = Integer.toString(p);
         out.writeUTF(pstr); // Sending p

         gstr = Integer.toString(g);
         out.writeUTF(gstr); // Sending g

         double A = ((Math.pow(g, a)) % p); // calculation of A
         Astr = Double.toString(A);
         out.writeUTF(Astr); // Sending A

         // Client's Private Key
         System.out.println("From Client : Private Key = " + a);

         // Accepts the data
         DataInputStream in = new
DataInputStream(client.getInputStream());

         serverB = Double.parseDouble(in.readUTF());
         System.out.println("From Server : Public Key = " +
serverB);

         Adash = ((Math.pow(serverB, a)) % p); // calculation
of Adash

         System.out.println("Secret Key to perform Symmetric
Encryption = "
                  + Adash);
         client.close();
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

OUTPUT:

1. SERVER PROGRAM



2. CLIENT PROGRAM



Experiment → 09

Aim → WAP to implement ElGamal encrypt^{n} algorithm.

Theory →

↳ ELGamal encryption is a public key cryptosystem. It uses assymmetric key encryption for comm^{n} between two parties and encrypting message.

↳ This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group that is even if we know g^a and g^k.

↳ Suppose Alice wants to communicate to Bob.

1. Bob generates public and private key.
2. From F_q, he choose any g.

3. Then he computes $h = g^a$.
4. BOB publishes F, $h = g^a$, q and g as his public key and retains "a" as private key.

2. Alice encrypts data using BOB's public key;

1. Alice selects r from F.
2. Then computes $p = g^r$ & $s = g^{ar}$
3. she multiplies 's' with 'm'.

3. BOB Decrypts message;

1. BOB calculates $s = p^a = g^{ar}$.
2. He divides m*s.

Conclusion →

Thus we have written a Java program to implement ElGamal Encryption algorithm.

```
/*
NAME: FURKHAN MUJIBODDEN SHAIKH
CLASS: BECSEII
AIM: Write a program to implement ElGamal encryption algorithm.
ROLL: 63
------------------------------------------------------------------------
*/
import java.math.*;
import java.util.*;
import java.security.*;
import java.io.*;

public class EXP9
{
   public static void main(String[] args) throws IOException
   {
      Scanner in = new Scanner(System.in);
      BigInteger p, b, c, secretKey;
      Random sc = new SecureRandom();
      secretKey = new BigInteger("12345678901234567890");

      System.out.println("Secret Key = " + secretKey);
      p = BigInteger.probablePrime(64, sc);
      b = new BigInteger("3");
      c = b.modPow(secretKey, p);
      System.out.println("p = " + p);
      System.out.println("b = " + b);
      System.out.println("c = " + c);

      System.out.println("Enter your Big Number message ");
      String s = in.nextLine();
      BigInteger X = new BigInteger(s);
      BigInteger r = new BigInteger(64, sc);
      BigInteger EC = X.multiply(c.modPow(r, p)).mod(p);
      BigInteger brmodp = b.modPow(r, p);
      System.out.println("Plaintext = " + X);
      System.out.println("r = " + r);
      System.out.println("EC = " + EC);
      System.out.println("b^r mod p = " + brmodp);
      BigInteger crmodp = brmodp.modPow(secretKey, p);
      BigInteger d = crmodp.modInverse(p);
      BigInteger ad = d.multiply(EC).mod(p);
      System.out.println("\n\nc^r mod p = " + crmodp);
      System.out.println("d = " + d);
      System.out.println("Decrypted Code is: " + ad);
      in.close();
   }
}
/*
OUTPUT:
Secret Key = 12345678901234567890
p = 11011430496307263017
b = 3
c = 762684688640510407

Enter your Big Number message
123456778
Plaintext = 123456778
r = 1466427748554375912
EC = 11001532579463450987
b^r mod p = 7637210775905307426

c^r mod p = 3743922534478032680
d = 4009754735560853414
Decrypted Code is: 123456778
*/
```

Experiment → 10

Aim → Write a Java pgm Study of digital certificate management in Web Browsers.

Theory →

↳ Digital certificate management allows you to manage digital certificates for your N/w and use Transport Layer security (TLS).

↳ The Digital certificate (DC) is an electronic credential that you can use to establish proof of identity in an electronic transaction.

There are an increasing number of uses for digital certificates to provide enhanced network security measures.

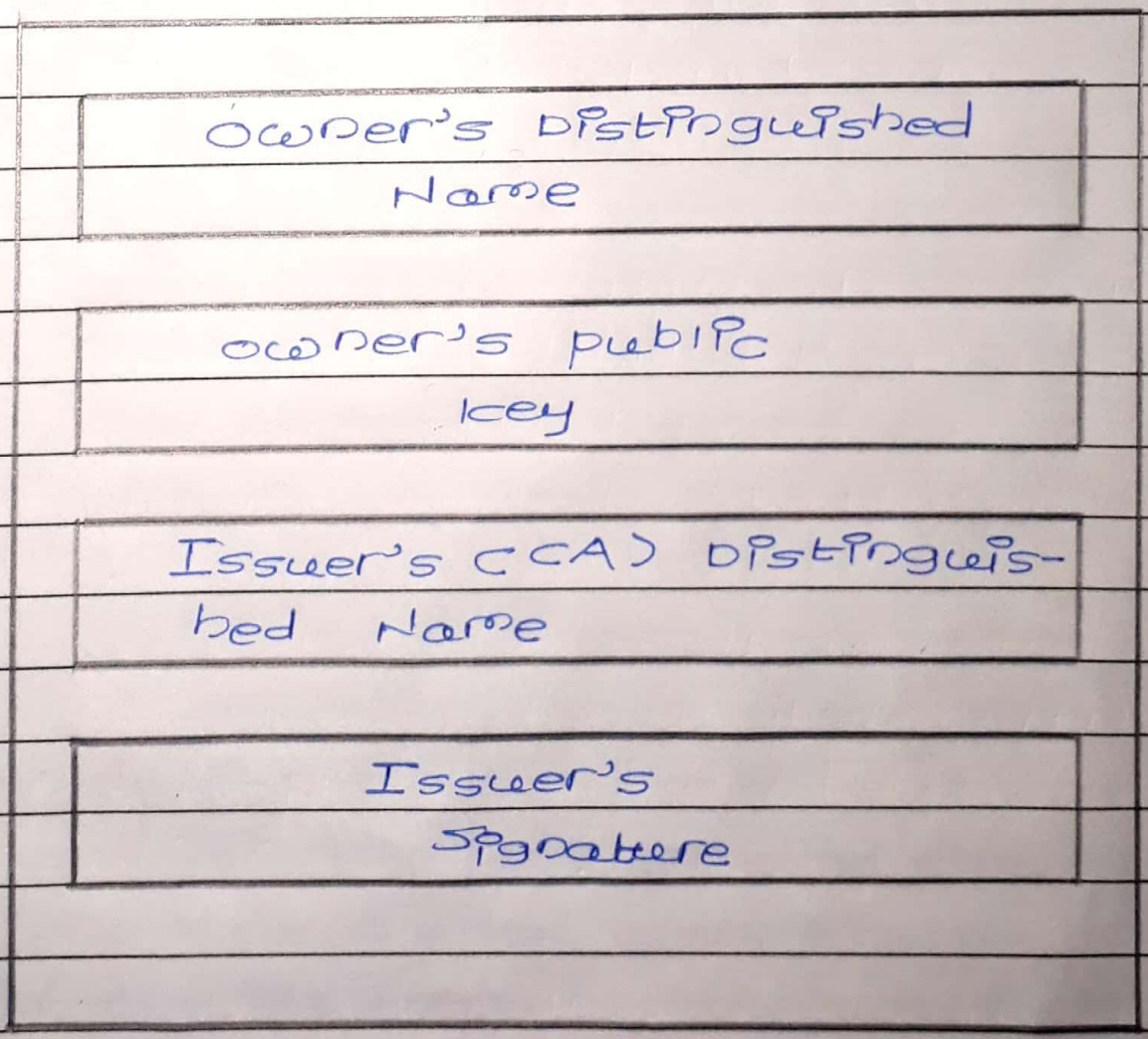
↳ IBM provides digital certificate support.

PDF file for DCM:

You can view and print a PDF file of this information.

DCM concepts:

A digital certificate is a digital credential for owner identity.



Conclusion →

So, We have studied how to manage digital certificates in Web browsers.