EX.NO:1.1 # CAESAR CIPHER

DATE:

AIM:

To write a java program to perform encryption and decryption using Caesar cipher algorithm.

ALGORITHM:

- Read the plaintext
- Invoke the method for encryption
- The Caesar cipher encryption involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.
    - C: E(P,3) = (P + 3) mod 26
- Invoke decryption method
    - P: D(C, 3) = (C + 3) mod 26

PROGRAM:

```
import java.util.Scanner;
  class cipher
{
    String alphabet = "abcdefghijklmnopqrstuvwxyz";
     String encrypt(String plainText, int shiftKey)
    {
        plainText = plainText.toLowerCase();
        String cipherText = "";
        for (int i = 0; i < plainText.length(); i++)
        {
            int charPosition =
alphabet.indexOf(plainText.charAt(i));
            int keyVal = (shiftKey + charPosition) % 26;
            char replaceVal = alphabet.charAt(keyVal);
            cipherText += replaceVal;
        }
        return cipherText;
    }

    String decrypt(String cipherText, int shiftKey)
    {
        cipherText = cipherText.toLowerCase();
        String plainText = "";
        for (int i = 0; i < cipherText.length(); i++)
        {
            int charPosition =
alphabet.indexOf(cipherText.charAt(i));
            int keyVal = (charPosition - shiftKey) % 26;
            if (keyVal < 0)
            {
                keyVal = alphabet.length() + keyVal;
```

```
            }
            char replaceVal = alphabet.charAt(keyVal);
            plainText += replaceVal;
        }
        return plainText;
    }
}
class Ccipher
{
    public static void main(String[] args)
    {
       cipher c=new  cipher();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the String for Encryption: ");
        String message = new String();
        message = sc.next();
       String cipherText = new String();
        cipherText=c.encrypt(message, 3);
        System.out.println("Encryption:");
        System.out.println(cipherText);
        System.out.println("Decryption:");
        System.out.println(c.decrypt(cipherText, 3));
        sc.close();
    }
}
```

OUTPUT:

D:\Java\jdk1.6.0\bin>javac Ccipher.java

D:\Java\jdk1.6.0\bin>java Ccipher

Enter the String for Encryption:

computer

Encryption:

frpsxwhu

Decryption:

Computer

RESULT:

Thus a java program to perform encryption and decryption using Caesar cipher algorithm was executed successfully.

EX.NO: 1.2

PLAYFAIR CIPHER

DATE:

AIM:

To write a java program to perform encryption and decryption using Playfair cipher technique.

ALGORITHM:

- The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.
- Plaintext is encrypted two letters at a time, according to the following rules:
- Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x.
- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.
- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last.
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

PROGRAM:

```
import java.util.*;
class Playfaircipher1
{
 public static void main(String[] args)
  {
   int j,i,m=0,k=0,val,z=0,flag=0,count=0;
   char T[][]=new char[5][5];
   String enmsg="";
   String plainmsg="";
   String alpha="abcdefghiklmnopqrstuvwxyz";
   Scanner sc=new Scanner(System.in);
   System.out.println("\n\nenter the msg:\t");
   String msg=sc.next();
   System.out.println("\n\nenter the key:\t");
   String key=sc.next();
  char p[]=new char[26];
  for(i=0;i<alpha.length();i++)
  {
   for(j=0;j<key.length();j++)
   {
    if(alpha.charAt(i)==key.charAt(j))
    {
     flag=1;
```

```
      break;
     }
    }
    if(flag==0)
    {
     p[z]=alpha.charAt(i);
     z++;
    }
    flag=0;
   }
  z=0;
   for(i=0;i<5;i++)
   {
    for(j=0;j<5;j++)
    {
     if(count==key.length())
     {
      T[i][j]=p[z];
      z++;
     }
     else
     {
      T[i][j]=key.charAt(m);
      m++;
      count++;
     }
    }
   }
            System.out.println("\n\nThe matrix:\n");
            for(i=0;i<5;i++)
            {
             for(j=0;j<5;j++)
             {
              System.out.print(T[i][j]);
              System.out.print("\t");
             }
             System.out.println("\n");
            }
val=msg.length();
if((val%2)==1)
msg+='x';
int I1=0,I2=0,J1=0,J2=0,c,flag1=0,flag2=0;
m=0;
   //encryption
  do
  {
   for(i=0;i<5;i++)
   {

        for(j=0;j<5;j++)
```

```
      {
           if(T[i][j]==msg.charAt(m))
            {
             I1=i;
             J1=j;
             flag1=1;
            }
           if(T[i][j]==msg.charAt(m+1))
            {
             I2=i;
             J2=j;
             flag2=1;
             }
           if(flag1==1 && flag2==1)
             break;
       }
if(flag1==1 && flag2==1)
      {
           if(I1==I2)
              c=1;
           else if(J1==J2)
              c=2;
           else
              c=3;
             switch(c)
               {
                   case 1:

                          if((J1+1)==5)
                             enmsg+=T[I1][0];
                          else
                             enmsg+=T[I1][J1+1];

                           if((J2+1)==5)
                             enmsg+=T[I2][0];
                           else
                             enmsg+=T[I2][J2+1];
                           break;
                     case 2:
                           if((I1+1)==5)
                               enmsg+=T[0][J1];
                           else
                               enmsg+=T[I1+1][J1];
                           if((I2+1)==5)
                                enmsg+=T[0][J2];
                           else
                                enmsg+=T[I2+1][J2];
                           break;

                  case 3:
```

```
                                    enmsg+=T[I1][J2];
                                    enmsg+=T[I2][J1];
                                    break;
                        }//switch end
          break;
         }//if end
       }//i loop end
 m=m+2;
 flag1=0;
 flag2=0;
 }while(m<msg.length());
System.out.println("\n\nPlayfair Cipher Text:\n\t"+ enmsg);
m=0;
//decryption
flag1=0;
 flag2=0;
do
 {
   for(i=0;i<5;i++)
   {
        for(j=0;j<5;j++)
         {
              if(T[i][j]==enmsg.charAt(m))
               {
                I1=i;
                J1=j;
                flag1=1;
              }
              if(T[i][j]==enmsg.charAt(m+1))
               {
                I2=i;
                J2=j;
                flag2=1;
                }
              if(flag1==1 && flag2==1)
                break;
         }
    if(flag1==1 && flag2==1)
         {
              if(I1==I2)
               c=1;
              else if(J1==J2)
                 c=2;
              else
                 c=3;
            switch(c)
                 {
                     case 1:
                             if(J1==0)
                              plainmsg+=T[I1][4];
```

```
                                    else
                                       plainmsg+=T[I1][J1-1];
                                     if(J2==0)
                                     plainmsg+=T[I2][4];
                                     else
                                    plainmsg+=T[I2][J2-1];
                                     break;
                                   case 2:
                                      if(I1==0)
                                       plainmsg+=T[4][J1];
                                      else
                                          enmsg+=T[I1-1][J1];
                                      if(I2==0)
                                          plainmsg+=T[4][J2];
                                      else
                                           plainmsg+=T[I2-1][J2];
                                      break;
                        case 3:
                                    plainmsg+=T[I1][J2];
                                    plainmsg+=T[I2][J1];
                                      break;
                        }//switch end
      break;
        }//if end
   }//i loop end
 m=m+2;
 flag1=0;
 flag2=0;
 }while(m<enmsg.length());
System.out.println("\n\nPlayfair Plain Text:\n\t"+ plainmsg);
}
}
```

OUTPUT:

D:\Java\jdk1.6.0\bin>javac Playfaircipher1.java

D:\Java\jdk1.6.0\bin>java Playfaircipher1

enter the msg:

cryptography

enter the key:

security

The matrix:

s e c u r

i t y a b

d f g h k

l m n o p

q v w x z

Playfair Cipher Text:

usbnamkcboga

Playfair Plain Text:

cryptography

RESULT:

Thus a java program to perform encryption and decryption using Playfair cipher algorithm was executed successfully.

HILL CIPHER

EX.NO: 1.3

DATE:

AIM:

To write a java program to perform encryption using hill cipher algorithm.

ALGORITHM:

This encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters

$$\mathbf{C} = PK \bmod 26$$

$$(c1\ c2\ c3) = (p1\ p2\ p3)\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

$$c1 = (k_{11}p_1 + k_{21}p_2 + k_{31}p_3) \bmod 26$$
$$c2 = (k_{12}p_1 + k_{22}p_2 + k_{32}p_3) \bmod 26$$
$$c3 = (k_{13}p_1 + k_{23}p_2 + k_{33}p_3) \bmod 26$$

$$P = D(K, C) = CK^{-1} \bmod 26 = PKK^{-1} = P$$

PROGRAM:

```
import java.io.*;
import java.lang.*;
public class hillcipher
{
public static void main(String []arg)throws Exception
{
    int k[][]={{17,17,5}, {21,18,21}, {2,2,19}};
    int p[]=new int[300];
    int c[]=new int[300];
    int i=0;
    BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
     System.out.println("enter plain text");
    String str=br.readLine();
     for( i=0;i<str.length();i++)
    {
        int c1=str.charAt(i);
       p[i]=(c1)-97;
    }
          i=0;int zz=0;
        for( int b=0;b<str.length()/3;b++)
        {
```

```
        for(int j=0;j<3;j++)
        {
        for(int x=0;x<3;x++)
        {
        c[i]+=k[j][x]*p[x+zz];
        }i++;
        }
        zz+=3;
        }
                System.out.println("Encrypted Text : ");
        for(int z=0;z<str.length();z++)
        System.out.print((char)((c[z]%26)+97));
 }
}
```

OUTPUT:

C:\Java\jdk1.6.0\bin>javac hillcipher.java

C:\Java\jdk1.6.0\bin>java hillcipher

enter plain text

paymoremoney

Encrypted Text :

lnshdlewmtrw

RESULT:

Thus a java program to perform encryption using hill cipher algorithm was executed successfully.

EX.NO: 1.4 **VIGENERE CIPHER**

DATE:

AIM:

To write a java program to perform encryption and decryption using vigenere cipher technique.

ALGORITHM:

- Read the plaintext text and keyword
- To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword.
- The first letter of the key is added to the first letter of the plaintext, mod 26,the second letters are added, and so on through the first m letters of the plaintext.
- Encryption process
  - $C_i = (p_i + k_{i \bmod m}) \bmod 26$
- Decryption process
  - $p_i = (C_i - k_{i \bmod m}) \bmod 26$

PROGRAM:

```
import java.util.*;
class Vigeneree
{
  public static void main(String[] args)
  {
    Scanner sc=new Scanner(System.in);
    String alpha="abcdefghijklmnopqrstuvwxyz";
    char T[][]=new char[26][26];
    String key=new String();
    String msg=new String();
    System.out.println("\nEnter the key and plaintext");
    key=sc.next();
    msg=sc.next();
    String enmsg="";
    String plmsg="";
    int i,j,count,m=0,h=1;
    for(i=0;i<26;i++)
    {
     count=i;
     for(j=0;j<26;j++)
     {
       T[i][j]=alpha.charAt(count);
       count=(count+1)%26;
     }
    }
          //key generation
```

```
  int len=msg.length();
  int klen=key.length();
  count=0;
  while(klen<len)
  {
   key+=key.charAt(count);
   count++;
   klen++;
  }
  System.out.println("Message   :\t"+msg);
  System.out.println("Key Text  :\t"+key);
          //encryption
  while(m<len)
  {
    i=alpha.indexOf(key.charAt(m));
    j=alpha.indexOf(msg.charAt(m));
    enmsg+=T[i][j];
    m++;
  }
  System.out.println("Cipher Text:\t"+enmsg);
          //decryption
  m=0;
  while(m<len)
  {
   i=0;
   j=alpha.indexOf(key.charAt(m));
   char k=enmsg.charAt(m);
   while(h==1)
   {
    if(T[i][j]==k)
    break;
    else
    i++;
   }
   plmsg+=alpha.charAt(i);
   m++;
  }
   System.out.println("Plain Text:\t"+plmsg);
}
 }
```

OUTPUT:

```
C:\Java\jdk1.6.0\bin>javac Vigenere.java
C:\Java\jdk1.6.0\bin>java Vigenere
Enter the key and plaintext
deceptive
wearediscoveredsaveyourself
Message   :     wearediscoveredsaveyourself
Key Text  :     deceptivedeceptivedeceptive
Cipher Text:    zicvtwqngrzgvtwavzhcqyglmgj
Plain Text:     wearediscoveredsaveyourself
```

RESULT:

Thus a java program to perform encryption and decryption using vigenere cipher technique was executed successfully.

EX.NO:2.1

RAIL FENCE

DATE:

AIM:

To write a java program to perform encryption and decryption using rail fence cipher technique.

ALGORITHM:

- Read the plaintext
- The plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.
- Read the character array, the characters at even positions are stored in an array and characters at odd positions are stored in another array.
- Both the arrays are concatenated and displayed

PROGRAM:

```
import java.util.*;
class Rail
{
  public static void main(String[] args)
  {
  Scanner sc=new Scanner(System.in);
  String msg=new String();
  System.out.println("Enter the message:");
  msg=sc.next();
  int i,ptr;
  char a[]=new char[20];
  char b[]=new char[20];
  String enmsg="";
  String pltxt="";
   i=0;
  for(ptr=0;ptr<msg.length();ptr=ptr+2)
  {
  a[i]=msg.charAt(ptr);
  i++;
  }
    i=0;
  for(ptr=1;ptr<msg.length();ptr=ptr+2)
  {
  b[i]=msg.charAt(ptr);
  i++;
  }
  i=0;
  for(ptr=0;ptr<msg.length()/2;ptr++)
```

```
  {
  enmsg+=a[i];
  i++;
  }
  i=0;
  for(ptr=0;ptr<msg.length()/2;ptr++)
  {
  enmsg+=b[i];
  i++;
  }
  System.out.println("Cipher Text:" +enmsg);
  i=0;
  for(ptr=0;ptr<msg.length()/2;ptr++)
  {
  pltxt+=a[i];
  pltxt+=b[i];
  i++;
  }
  System.out.println("Plain Text:" +pltxt);
  }
}
```

OUTPUT

C:\Java\jdk1.6.0\bin>javac Rail.java

C:\Java\jdk1.6.0\bin>java Rail

Enter the message:

meetmeafterthetogaparty

Cipher Text:mematrhtgpretefeteoaat

Plain Text:meetmeafterthetogapart

RESULT:

Thus a java program to perform encryption and decryption using rail fence cipher technique was executed successfully.

EX.NO:2.2 # ROW COLUMN TRANSPOSITION

DATE:

AIM:

To write a java program to perform encryption and decryption using row column transposition technique.

ALGORITHM:

- Read the plaintext
- The plaintext is written down as a sequence of row by row
- Then read in the sequence column by column.
- The Key is used to read the column order
- The read character array is displayed as encrypted message.
- Decrypt the encrypted message with the same key.
- Display the decrypted message.

PROGRAM:

```
import java.util.Scanner;
public class ColTransCipher {
   private static Scanner in;
   public static void main(String[] args){
      System.out.println("Columnar Transposition Cipher");
      in = new Scanner(System.in);
      System.out.print("1. Encryption\n2. Decryption\nChoose(1,2): ");
      int choice = in.nextInt();
      in.nextLine();
      if (choice == 1){
         System.out.println("Encryption");
         encryption();
      } else if (choice == 2){
         System.out.println("Decryption");
         decryption();
         System.out.println("Invalid Choice");
         System.exit(0);
      }
   }
   private static void encryption(){
      System.out.print("Enter Message: ");
```

```
        String plainText = in.nextLine().toUpperCase().replace(" ", "");
        StringBuilder msg = new StringBuilder(plainText);
        System.out.print("Enter Keyword: ");
        String keyword = in.nextLine().toUpperCase();
        int[] kywrdNumList = keywordNumAssign(keyword);
        for (int i = 0, j = 1; i < keyword.length(); i++, j++) {
            System.out.print(keyword.substring(i, j) + " ");
        }
        System.out.println();

        for (int i: kywrdNumList){
            System.out.print(i + " ");
        }
        System.out.println();
        System.out.println("-------------------------");
        int extraLetters = msg.length() % keyword.length();
        //System.out.println(extraLetters);
        int dummyCharacters = keyword.length() - extraLetters;
//          System.out.println(dummyCharacters);
        if (extraLetters != 0){
            for (int i = 0; i < dummyCharacters; i++) {
                msg.append(".");
            }
        }
        int numOfRows = msg.length() / keyword.length();
        char[][] arr = new char[numOfRows][keyword.length()];
        int z = 0;
        for (int i = 0; i < numOfRows; i++) {
            for (int j = 0; j < keyword.length(); j++) {
                arr[i][j] = msg.charAt(z);
                z++;
        }

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

```
    StringBuilder cipherText = new StringBuilder();
    System.out.println();
    String numLoc = getNumberLocation(keyword, kywrdNumList);
    System.out.println("Location of numbers: " + numLoc);
    System.out.println();
    for (int i = 0, k = 0; i < numOfRows; i++, k++) {
        int d;
        if (k == keyword.length()){
            break;
        } else {
            d = Character.getNumericValue(numLoc.charAt(k));
        }
        for (int j = 0; j < numOfRows; j++) {
            cipherText.append(arr[j][d]);
        }
    }
    System.out.println("Cipher Text: " + cipherText);
}
private static void decryption(){
    System.out.print("Enter Message: ");
    String msg = in.nextLine().toUpperCase().replace(" ", "");
    System.out.print("Enter Keyword: ");
    String keyword = in.nextLine().toUpperCase();
    int numOfRows = msg.length() / keyword.length();
    char[][] arr = new char[numOfRows][keyword.length()];
    int[] kywrdNumList = keywordNumAssign(keyword);
    String numLoc = getNumberLocation(keyword, kywrdNumList);
    for (int i = 0, k = 0; i < msg.length(); i++, k++) {
        int d = 0;
        if (k == keyword.length()){
            k = 0;
        } else {
            d = Character.getNumericValue(numLoc.charAt(k));
        }

        for (int j = 0; j < numOfRows; j++, i++) {
            arr[j][d] = msg.charAt(i);
        } // for loop
        --i;
    }
```

```
        StringBuilder plainText = new StringBuilder();

        for (int i = 0; i < numOfRows; i++) {
            for (int j = 0; j < keyword.length(); j++) {
                plainText.append(arr[i][j]);
            }
        }
        System.out.println("Plain Text: " + plainText);
    }
    private static String getNumberLocation(String keyword, int[]
kywrdNumList) {
        String numLoc = "";
        for (int i = 1; i < keyword.length() + 1; i++) {
            for (int j = 0; j < keyword.length(); j++) {
                if (kywrdNumList[j] == i){
                    numLoc += j;
                }
            }
        }
        return numLoc;
    }
    private static int[] keywordNumAssign(String keyword) {
        String alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        int[] kywrdNumList = new int[keyword.length()];

        int init = 0;
        for (int i = 0; i < alpha.length(); i ++){
            for (int j = 0; j < keyword.length(); j++) {
                if (alpha.charAt(i) == keyword.charAt(j)){
                    init++;
                    kywrdNumList[j] = init;
                }
            }
        }
        return kywrdNumList;
    }
```

OUTPUT :

C:\Users\placement\Desktop\New folder (2)>java ColTransCipher

Columnar Transposition Cipher

1. Encryption

2. Decryption

Choose(1,2): 1

Encryption

Enter Message: I LIKE POTATOES BECAUSE THEY ARE TASTY

Enter Keyword: POTATO

P O T A T O

4 2 5 1 6 3

I L I K E P

O T A T O E

S B E C A U

S E T H E Y

A R E T A S

T Y . . . .

Location of numbers: 315024

Cipher Text: KTCHT.LTBERYPEUYS.IOSSATIAETE.EOAEA.

C:\Users\placement\Desktop\New folder (2)>java ColTransCipher

Columnar Transposition Cipher

1. Encryption

2. Decryption

Choose(1,2): 2

Decryption

Enter Message: KTCHT.LTBERYPEUYS.IOSSATIAETE.EOAEA.

Enter Keyword: POTATO

Plain Text: ILIKEPOTATOESBECAUSETHEYARETASTY....

RESULT :

Thus a java program to perform encryption and decryption using Row Column Transposition technique has been executed successfully.

EX.NO:3 **DES**

DATE:

AIM:

To write a java program to perform encryption using Data Encryption Standard (DES) .algorithm.

ALGORITHM:

- Initial permutation rearranging the bits to form the "permuted input".
- Followed by 16 iterations of the same function (substitution and permutation).
- The output of the last iteration consists of 64 bits which is a function of the plaintext and key. The left and right halves are swapped to produce the preoutput.
- Finally, the preoutput is passed through a permutation which is simply the inverse of the initial permutation (IP). The output of IP^{-1} is the 64-bit ciphertext.
- Initially the key is passed through a permutation function
- For each of the 16 iterations, a subkey ($\mathbf{K}_i$) is produced by a combination of a left circular shift and a permutation which is the same for each iteration. However, the resulting sub key is different for each iteration because of repeated shifts

PROGRAM:

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
public class DESS
 {
 private static Cipher encryptCipher;
 private static Cipher decryptCipher;
 public static void main(String[] args)
{
  try {
   KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
   SecretKey secretKey = keygenerator.generateKey();
   encryptCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
   encryptCipher.init(Cipher.ENCRYPT_MODE, secretKey);
   byte[] encryptedData = encryptData("Classified Information!");
   decryptCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
   decryptCipher.init(Cipher.DECRYPT_MODE, secretKey);
   decryptData(encryptedData);
```

```
  }
catch (NoSuchAlgorithmException e)
{
   e.printStackTrace();
  }
catch (NoSuchPaddingException e)
{
   e.printStackTrace();
  }
catch (InvalidKeyException e) {
   e.printStackTrace();
  }
catch (IllegalBlockSizeException e) {
   e.printStackTrace();
  }
catch (BadPaddingException e) {
   e.printStackTrace();
  }
 }
 // Encrypt Data
 private static byte[] encryptData(String data)   throws
IllegalBlockSizeException, BadPaddingException {
  System.out.println("Data Before Encryption :" + data);
  byte[] dataToEncrypt = data.getBytes();
  byte[] encryptedData = encryptCipher.doFinal(dataToEncrypt);
  System.out.println("Encryted Data: " + encryptedData);
  return encryptedData;
 }
 // Decrypt Data
   private static void decryptData(byte[] data)   throws
IllegalBlockSizeException, BadPaddingException
{
  byte[] textDecrypted = decryptCipher.doFinal(data);
  System.out.println("Decryted Data: " + new
String(textDecrypted));
 }
}
```

OUTPUT:

```
D:\Java\jdk1.6.0\bin>java DESS
Data Before Encryption :Classified Information!
Encryted Data: [B@4e79f1
Decryted Data: Classified Information!
```

RESULT:

Thus a java program to perform encryption using Data Encryption Standard(DES) algorithm was executed successfully.

EX.NO: 4 # IMPLEMENTATION OF AES IN JAVA

Date:

AIM:

To write a Java program to implement AES Algorithm.

ALGORITHM:

AES steps of encryption:

1. Derive the set of round keys from the cipher key.
2. Initialize the state array with the block data (plaintext).
3. Add the initial round key to the starting state array.
4. Perform nine rounds of state manipulation.
5. Perform the tenth and final round of state manipulation.
6. Copy the final state array out as the encrypted data (ciphertext).

PROGRAM:

```
package com.javapapers.java.security;

import java.util.Base64;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

public class EncryptionDecryptionAES {

    static Cipher cipher;

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

        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");

        keyGenerator.init(128);
```

```
        SecretKey secretKey = keyGenerator.generateKey();
        cipher = Cipher.getInstance("AES");
        String plainText = "AES Symmetric Encryption Decryption";
        System.out.println("Plain Text Before Encryption: " + plainText);
        String encryptedText = encrypt(plainText, secretKey);
        System.out.println("Encrypted Text After Encryption: " + encryptedText);
        String decryptedText = decrypt(encryptedText, secretKey);
        System.out.println("Decrypted Text After Decryption: " + decryptedText);
}
public static String encrypt(String plainText, SecretKey secretKey)
                throws Exception {
        byte[] plainTextByte = plainText.getBytes();
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedByte = cipher.doFinal(plainTextByte);
        Base64.Encoder encoder = Base64.getEncoder();
        String encryptedText = encoder.encodeToString(encryptedByte);
        return encryptedText;
}
public static String decrypt(String encryptedText, SecretKey secretKey)
                throws Exception {
        Base64.Decoder decoder = Base64.getDecoder();
        byte[] encryptedTextByte = decoder.decode(encryptedText);
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedByte = cipher.doFinal(encryptedTextByte);
        String decryptedText = new String(decryptedByte);
```

```
        return decryptedText;
    }
}
```

OUTPUT:

```
Plain Text Before Encryption: AES Symmetric Encryption Decryption

Encrypted Text After Encryption:
sY6vkQrWRg0fvRzbqSAYxepeBIXg4AySj7Xh3x4vDv8TBTkNiTfca7wW/dxiMMJl

Decrypted Text After Decryption: AES Symmetric Encryption Decryption
```

RESULT:

Thus a Java program to implement AES Algorithm was executed successfully.