Exno: (1A)

Encryption and Decryption Using Caesar Cipher

AIM:

To encrypt and decrypt the given message by using Caesar Cipher encryption algorithm.

ALGORITHM:

Step 1: In Caesar cipher each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.

Step 2: For example, with a left shift of 3, D would be replaced by A, E would become B, and so on.

Step 3: The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the Scheme, $A=0, B=1, Z=25$

Step 4: Encryption of a letter x by a shift n can be described mathematically as, $En(x) = (x+n) \bmod 26$.

Step 5: Decryption is performed similarly, $Dn(x) = (x-n) \bmod 26$

Program:

Caesar cipher.java.

```
class caesar Cipher {
  public static string encode (string enc, int offset){
        offset = offset % 26 + 26;
    String Builder encoded = new StringBuilder();
    for (char i : enc to char Array ()){
          if (character . isLetter (i)) {
              if (character . is UpperCase (i)) {
    encoded . append (char) ('A' + (i - 'A' + offset) % 26));
          } else
          {
```

Output:

Simulating Caesar Cipher
- - - - - - - - - - -

Input:

Encrypted Message :

Decrypted Message :

```
encoded.append((char) ('a' + (i-'a'+offset) % 26));
   }
 } else {
   encoded.append(i);
 }
 }
 return encoded.toString();
 }
 public static String decode (string enc, int offset){
   return encode (enc, 26-offset);
}
public static void main (string [] args) throws java.lang.Exception {
   String msg =                         ;
System.out.println ("Simulating Caesar Cipher \n - - - - -");
System.out.println ("Input :"+ msg);
System.out.println ("Encrypted Message :");
System.out.println (caesarCipher.encode (msg, 3));
System.out.println ("Decrypted Message :");
System.out.println (caesarCipher.decode (caesarCipher.encode
                                        (msg,3), 3));
}
}
```

Result:

Thus the program for Caesar Cipher encryption and decryption algorithm has been implemented and the output is Verified Successfully.

EXNO-(1B)

Playfair Cipher.

AIM:

To implement a program to encrypt a plain text and decrypt a Cipher text Using playfair Cipher Substitution technique.

ALGORITHM:

Step 1: To encrypt a message, one would break the message into diagrams (groups of 2 letters).

Step 2: For example, "Hello World" becomes "HE LL OW OR LD"

Step 3: These diagrams will be substituted using the Key table.

Step 4: Since encryption requires pairs of letters, messages with an odd number of characters usually append an Uncommon letter, Such as "X", to Complete the final diagram.

Step 5: The two letters of the diagram are Considered opposite Corners of a Rectangle in the Key table. To perform the Substitution, apply the following 4 rules, in order, to each pair of letters in the plain text.

Program:

play fair Cipher. Java

```
    import java. awt. point;
class playfair Cipher {
 private static char [][] char Table;
 private static point [] positions;
 private static String prepare Text (string s, boolean chgJtoI){
    S= S. to Upper (). replace All (" [^A-Z]", " ");
} return chg JtoI ? is-replaced ("J","I"): s. replace ("Q","");

 private static void create Tbl (String Key, boolean chg JtoI) {
    char Table  = new char[5][5];
    positions   = new point [26];
```

```
String s= prepare Text (Key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                              chgJtoI));
  int len= s.length();
  for (int i=0, K=0 ; i< len ; i++){
      char C = s.charAt(i);
  if(positions [c - 'A'] == null) {
    char Table [K/5][K%5] = C;
  positions [c-'A'] = new point (K%5, K/5);
      K++ ;
  }
  }
  }
private static string Codec(string Builder txt, int dir){
  int len= txt.length ();
for (int i=0; i<len; i+=2){
char a= txt.charAt(i);
char b = txt.charAt(i+1);
int row1 = positions [a-'A'].y;
int row2 = positions [b-'A'].y;
int col1 = positions [a - 'A'].x;
int col2 = positions [b- 'A'].x;
  if (row1 == row2) {
    col1= (col1 + dir) %5 ;
    col2 = (col2+dir)%5 ;
  } else if (col1 == col2) {
 row1 =(row1+dir)%5;
 row2 =(row2+dir)%5;
 } else
 {
 int temp = col1;
   col1 = col2;
   col2 = temp;
}
txt.set charAt(i, char Table [row1] [col 1]);
txt.set charAt (i+1, char Table [row2] [col2]);
}
return txt.to string();
}
private static string encode (string s) {
   String Builder sb = new String Builder(s);
for(int i=0 ; i<sb.length(); i+=2){
   if (i==sb.length()-1){
sb.append (sb.length() % 2 ==1 ? 'x' : ""); }
else if (sb.charAt(i) == sb.charAt (i+1)) {
  sb.insert (i+1, 'x');

}
}
return code c (sb,1);
}
private static string decode (string s) {
  return Codec (new string Builder (s), 4);
}
private static void main (string [] args) throws java.lang.
                                                    Exception {
String key =             ;
String txt =             ;
boolean chgJtoI = true;
Create Tbl (key , chgJtoI);
String enc = encode (prepare Text (txt, chg JtoI));
System.out.println("Simulating playfair Cipher\n - - - -  ");
System.out.println ( "Input Message : " + txt );
System.out.println("Encrypted Message : "+ enc);
```

Output:

```
Simulating play fair Cipher
-----------
Input message :
Encrypted Message:
Decrypted Message:
```

```
System out . println ("Decrypted Message : "+ decode (enc));
}
}
```

Result:

Thus the program for play fair Cipher encryption and decryption algorithm has been implement and the Output is Verified Successfully .

Exno- 1C

Hill Cipher

AIM:

To implement a program to encrypt and decrypt using the Hill Cipher Substitution Technique.

ALGORITHM:

Step 1: In the Hill cipher Each letter is represented by a number modulo 26.

Step 2: To encrypt a msg, each block of n letters is multiplied by an invertible n xn matrix, again modulo 26.

Step 3: To decrypt the msg, each block is multiplied by the inverse of the matrix used for encryption.

Step 4: The matrix used for encryption is the Cipher Key and it should be chosen randomly from the set of invertible nxn matrices (modulo 26).

Step 5: The Cipher can, be adapted to an alphabet with any number of letters.

Step 6: All arithmetic just needs to be done modulo the number of letters instead of modulo 26.

Program:

Hill Cipher .java

```
class hillcipher {
 public static int [] [] Key mat = new int [] {{1,2,1},{2,3,2},
                              {2, 2, 1}};

 public static int [] [] inv Keymat = new int [] [] {{-1,0,1},
                         {2,-1,0} ,{-2,2,-1}};

 public static string Key = "ABCDEFGHIJ KLMNOPQRST UV
                                        WXYZ";
```

```
private static string encode (chara, charb, charc) {
  string ret = "  ";
  int x, y, z;
int pos a = (int) a - 65;
int pos b = (int) b - 65;
Int posc = (int) c - 65;
x = pos a * Key mat [0][0] + pos b * Keymat [1][0] + posc *
                                Keymat [2][0];
y = pos a * Keymat [0][1] + posb * Keymat [1][1] + posc *
                                Key mat [2][1];
z = pos a * Keymat [0][2] + pos b * Keymat [1][2] + posc *
                                Keymat [2][2];
a = Key.charAt (x % 26);
b = Key.charAt (y % 26);
c = Key.charAt (z % 26);
ret = "  " + a + b + c;
return ret;
}
private static string decode (char a, charb, charc) {
 String ret = "  ";
 int x, y, z
int pos a = (int) a - 65;
int pos b = (int) b - 65;
int posc = (int) c - 65;
x = pos a * inv Keymat [0][0] + posb * invKeymat [1][0] +
        posc * inv Keymat [2][0];
y = pos a * inv Key mat [0][1] + posb * inv Keymat [1][1] +
        posc * inv Keymat [2][1];
z = pos a * inv Keymat [0][2] + posb * inv Key mat [1][2] +
        posc * inv Keymat [2][2];
a = Key.charAt((x%26 <0)?(26+x%26):(x%26));
b = Key.charAt((y%26<0)?(26+y%26):(y%26));
c = Key.charAt((z%26<0)?(26+z%26):(z%26));

ret = " "+a+b+c;
return ret;
}
public static void main (string [] args) throws java.lang.
                                              Exception {
string msg;
string enc = " ";
String dec = " ";
int n;
msg = ("Security Laboratory");
System.out.println ("Simulation of Hill Cipher \n --------");
System.out.println ("Input message :"+msg);
msg = msg.toUpperCase();
msg = msg.replaceAll ("\\s", "");
   for( int i =1; i<=(3-n); i++){
      msg+ = 'x';
   }
   }
System.out.println ("padded message :"+ msg);
char [] pdchars = msg.toCharArray();
for(int i=0; i<msg.Length(); i+=3) {
    enc + = encode(pdchars [i], pdchars[i+1], pdchars[i+2]);
}
System.out.println("encoded message : "+enc);
```

Output:

Simulating Hill Cipher

```
-------------------
```

Input message :

Padded message :

Encrypted message:

Decrypted message:

```
char [] pd chars = msg. to Char Array ();
char[] de chars = enc. to char Array ();
  for (int i=0; i< enc . length (); i+=3) {
dec += decode (dechars [i], dechars [i+1], dechars [i+2]);
}
System . out . println ("decoded message : "+dec);
}
}
```

Result:

Thus the program for hill Cipher encryption and decryption has been implemented and the output Verified Successfully.

no-(10)

Vigenere Cipher.

AIM:

To implement a program for encryption and decryption using Vigenere Cipher Substitution technique.

ALGORITHM:

Step1: The Vigenere Cipher is a method of encrypting alphabetic text by using a series of different Caesar Ciphers based on the letters on the keyword.

Step2: It is a simple form of polyalphabetic substitution

Step3: To encrypt, a table of alphabets can be used, termed a Vigenere Square, or Vigenere table.

Step4: It Consists of the alphabet written out 26 times, in different rows, each alphabetic shifted Cyclically to the left Compared to the previous alphabet, Corresponding to the 26 possible Caesar objects.

Step5: At different points in the encryption process, the Cipher Uses a different alphabet from one of the rows used.

Step6: The alphabet at each point depends on a repeating Keyword.

Program:

Vigenere Cipher.java

```
public class vigenere Cipher {
  static string encode (string text, final string key){
     string res = " ";
     text = text. toUpperCase();
  for (int i=0, j=0; i<text.length(); i++) {
     char c = text.charAt(i);
       if((c<'A'|| c>'Z'){
```

Output:

```
Simulating Vigenere Cipher
-------
Input Message:
Encrypted Message:
Decrypted Message:
```

```
continue;
}
res+ = (char)((c+Key.charAt(j)-2*'A')%26+'A');
 j = ++j % Key.length();
}
return res;
}
static String decode (String text, final String Key){
    String res = " ";
    text = text.to UpperCase();
for(int i=0, j=0; i< text.length(); i++){
 char c = text.charAt(i);
if(c<'A' || c>'Z'){
   Continue;
 }
res+ = (char)((c-Key.charAt(j)+26)%26+'A');
j = ++j % Key.length();
}
return res;
}
public static void main (String [] args)throws java.lang.
                                         Exception {.
String Key =                           ;
String msg =                           ;
System.out.println ("Simulating Vigenere Cipher \n ------");
System.out.println ("Input Message : "+Msg);
String enc = encode (msg, Key);
System.out.println (" Encrypted Message : "+enc);
System.out.println ("Decrypted Message : "+decode (enc, Key));
}}
```

Result:

Thus the program for Vigenere Cipher encryption and decryption algorithm has been implemented and Output is Verified