# 3)a)CEASER CIPHER

# Code:-

```
import java.util.Scanner;
public class CaesarCipherExample
{
 public static final String ALPHABET ="abcdefghijklmnopqrstuvwxyz";
  public static String encryptData(String inputStr, int shiftKey)
  {
    inputStr = inputStr.toLowerCase();
    String encryptStr = "";
 for (int i = 0; i < inputStr.length(); i++)
 {
 // get position of each character of inputStr in ALPHABET
 int pos = ALPHABET.indexOf(inputStr.charAt(i));

 // get encrypted char for each char of inputStr
 int encryptPos = (shiftKey + pos) % 26;
 char encryptChar = ALPHABET.charAt(encryptPos);

 // add encrypted char to encrypted string
 encryptStr += encryptChar;
 }

 // return encrypted string
 return encryptStr;
 }

 // create decryptData() method for decrypting user input string with given shift key
 public static String decryptData(String inputStr, int shiftKey)
 {
```

```
// convert inputStr into lower case
inputStr = inputStr.toLowerCase();

// decryptStr to store decrypted data
String decryptStr = "";

// use for loop for traversing each character of the input string
for (int i = 0; i < inputStr.length(); i++)
{

// get position of each character of inputStr in ALPHABET
int pos = ALPHABET.indexOf(inputStr.charAt(i));

// get decrypted char for each char of inputStr
int decryptPos = (pos - shiftKey) % 26;

// if decryptPos is negative
if (decryptPos < 0){
decryptPos = ALPHABET.length() + decryptPos;
}
char decryptChar = ALPHABET.charAt(decryptPos);

// add decrypted char to decrypted string
decryptStr += decryptChar;
}
// return decrypted string
return decryptStr;
}

// main() method start
public static void main(String[] args)
```

```
{
 // create an instance of Scanner class
 Scanner sc = new Scanner(System.in);

 // take input from the user
 System.out.println("Enter a string for encryption using Caesar Cipher: ");
 String inputStr = sc.nextLine();

 System.out.println("Enter the value by which each character in the plaintext message gets
shifted: ");
 int shiftKey = Integer.valueOf(sc.nextLine());

 System.out.println("Encrypted Data ===> "+encryptData(inputStr, shiftKey));
 System.out.println("Decrypted Data ===> "+decryptData (encryptData(inputStr, shiftKey),
shiftKey));

 // close Scanner class object
 sc.close();
 }
}
```

## Output:-

```
Enter a string for encryption using Caesar Cipher:
hello
Enter the value by which each character in the plaintext message gets shifted:
3
Encrypted Data ===> khoor
Decrypted Data ===> hello
```

## 3)c)HILL CIPHER:-

## Code:-

```
class GFG
{
// Following function generates the
// key matrix for the key string
static void getKeyMatrix(String key, int keyMatrix[][])
{
int k = 0;
for (int i = 0; i < 3; i++)
{
for (int j = 0; j < 3; j++)
{
keyMatrix[i][j] = (key.charAt(k)) % 65;
k++;
}
}
}
// Following function encrypts the message
static void encrypt(int cipherMatrix[][],
int keyMatrix[][],
int messageVector[][])
{
int x, i, j;
for (i = 0; i < 3; i++)
{
for (j = 0; j < 1; 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;
}
}
}
// Function to implement Hill Cipher
static void HillCipher(String message, String key)
{
// Get key matrix from the key string
int [][]keyMatrix = new int[3][3];
getKeyMatrix(key, keyMatrix);
int [][]messageVector = new int[3][1];
// Generate vector for the message
for (int i = 0; i < 3; i++)
messageVector[i][0] = (message.charAt(i)) % 65;
int [][]cipherMatrix = new int[3][1];
// Following function generates
// the encrypted vector
encrypt(cipherMatrix, keyMatrix, messageVector);
String CipherText="";
// Generate the encrypted text from
// the encrypted vector
for (int i = 0; i < 3; i++)
CipherText += (char)(cipherMatrix[i][0] + 65);
// Finally print the ciphertext
System.out.print(" Ciphertext:" + CipherText);
}
// Driver code
public static void main(String[] args)
{
// Get the message to be encrypted
```

```
String message = "HEY";
// Get the key
String key = "GYBNQKURP";
HillCipher(message, key);
}
}
```

## Output:-

```
Ciphertext:GFW
```

## 4)DES ALGORITHM:-

## Code:-

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class Main {
private static byte[] toHexByteArray(String self) {
byte[] bytes = new byte[self.length() / 2];
for (int i = 0; i < bytes.length; ++i) {
int val = Integer.parseInt(self.substring(i * 2, i * 2 + 2), 16);
bytes[i] = (byte) val;
}
return bytes;
}
private static void printHexBytes(byte[] self, String label) {
System.out.printf("%s", label);
for (byte b : self) {
int bb = (b >= 0) ? ((int) b) : b + 256;
String ts = Integer.toString(bb, 16);
if (ts.length()< 2) {
ts = "0" + ts;
}
System.out.print(ts);
```

```
}
System.out.println();
}
public static void main(String[] args) throws Exception {

String strKey = "0e329232ea6d0d73";
byte[] keyBytes = toHexByteArray(strKey);
SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");

Cipher encCipher = Cipher.getInstance("DES");
encCipher.init(Cipher.ENCRYPT_MODE, key);

String strPlain = "8787878787878787";
byte[] plainBytes = toHexByteArray(strPlain); //string into byte conversion
byte[] encBytes = encCipher.doFinal(plainBytes);
printHexBytes(encBytes, "Encoded");
//DES Decryption
Cipher decCipher = Cipher.getInstance("DES");
decCipher.init(Cipher.DECRYPT_MODE, key);
byte[] decBytes = decCipher.doFinal(encBytes);
printHexBytes(decBytes, "Decoded");
}
}
```

## Output:-

```
Encoded0000000000000000a913f4cb0bd30f97
Decoded8787878787878787
```

## 5)BLOW FISH ALGORITHM:-

## Code:-

```
import java.io.UnsupportedEncodingException;
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
```

```
import java.util.Base64;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

public class BlowfishDemo {

   public String encrypt(String password, String key) throws
              NoSuchAlgorithmException, NoSuchPaddingException,
               InvalidKeyException, IllegalBlockSizeException,
                BadPaddingException, UnsupportedEncodingException {
      byte[] KeyData = key.getBytes();
      SecretKeySpec KS = new SecretKeySpec(KeyData, "Blowfish");
      Cipher cipher = Cipher.getInstance("Blowfish");
      cipher.init(Cipher.ENCRYPT_MODE, KS);
      String encryptedtext = Base64.getEncoder().
         encodeToString(cipher.doFinal(password.getBytes("UTF-8")));
      return encryptedtext;

   }

   public String decrypt(String encryptedtext, String key)
        throws NoSuchAlgorithmException, NoSuchPaddingException,
          InvalidKeyException, IllegalBlockSizeException,
                    BadPaddingException {
      byte[] KeyData = key.getBytes();
      SecretKeySpec KS = new SecretKeySpec(KeyData, "Blowfish");
      byte[] ecryptedtexttobytes = Base64.getDecoder().
                     decode(encryptedtext);
      Cipher cipher = Cipher.getInstance("Blowfish");
      cipher.init(Cipher.DECRYPT_MODE, KS);
      byte[] decrypted = cipher.doFinal(ecryptedtexttobytes);
      String decryptedString =
              new String(decrypted, Charset.forName("UTF-8"));
      return decryptedString;

   }

   public static void main(String[] args) throws Exception {
      final String password = "Aliet@123";
      final String key = "knowledgefactory";
      System.out.println("Password: " + password);
      BlowfishDemo obj = new BlowfishDemo();
      String enc_output = obj.encrypt(password, key);
      System.out.println("Encrypted text: " + enc_output);
      String dec_output = obj.decrypt(enc_output, key);
      System.out.println("Decrypted text: " + dec_output);
   }
}
```

## Output:-

```
Password: Aliet@123
```

Encrypted text: TnCkp+sDvJ8kb6fulS+NAQ==

Decrypted text: Aliet@123

## 6)Rijndael algorithm:-

## Code:-

```
import javax.crypto.*;
import javax.crypto.spec.*;
class Rijndael{

public static String  asHex  (byte  buf[])
{
   StringBuffer strbuf = new StringBuffer(buf.length * 2);
   int i;
for (i = 0;i < buf.length;i++)
{
   if (((int) buf[i] & 0xff) < 0x10)
strbuf.append("0");
strbuf.append(Long.toString((int) buf[i] & 0xff, 16));
}
return strbuf.toString();
}
public static void main(String[] args) throws Exception
{
   String message="cryptography lab!!";
KeyGenerator kgen = KeyGenerator.getInstance("AES");
kgen.init(128); // 192 and 256 bits may not be available
//  Generate  the  secret key  specs.
SecretKey skey = kgen.generateKey();
byte[] raw = skey.getEncoded();
SecretKeySpec skeySpec = new SecretKeySpec(raw,"AES");
// Instantiate the cipher
```

```
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal((args.length == 0 ? message : args[0]).getBytes());
System.out.println("encrypted string: " + asHex(encrypted));
cipher.init(Cipher.DECRYPT_MODE, skeySpec);
byte[] original = cipher.doFinal(encrypted);
String originalString = new String(original);
System.out.println("Original string: " + originalString + " " + asHex(original));
}
}
```

## Output:-

encrypted string:
f7f381f44f2d15cc35b8e7c3c8ecf22b5752682f00782019ec332c2b3d684119Original string:
cryptography lab!! 63727970746f677261706879206c61622121

## 8)RSA ALGORITHM:-

## Code:-

```
import java.math.*;
import java.util.*;
class RSA {
        public static void main(String args[])
        {
                int p, q, n, z, d = 0, e, i;

                // The number to be encrypted and decrypted
                int msg = 12;
                double c;
                BigInteger msgback;

                // 1st prime number p
                p = 3;
```

```
// 2nd prime number q
q = 11;
n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

        // e is for public key exponent
        if (gcd(e, z) == 1) {
                break;
        }
}
System.out.println("the value of e = " + e);
for (i = 0; i <= 9; i++) {
        int x = 1 + (i * z);

        // d is for private key exponent
        if (x % e == 0) {
                d = x / e;
                break;
        }
}
System.out.println("the value of d = " + d);
c = (Math.pow(msg, e)) % n;
System.out.println("Encrypted message is : " + c);

// converting int value of n to BigInteger
BigInteger N = BigInteger.valueOf(n);
```

```
                // converting float value of c to BigInteger
                BigInteger C = BigDecimal.valueOf(c).toBigInteger();
                msgback = (C.pow(d)).mod(N);
                System.out.println("Decrypted message is : "
                                                + msgback);
        }

        static int gcd(int e, int z)
        {
                if (e == 0)
                        return z;
                else
                        return gcd(z % e, e);
        }
}
```

## Output:-

```
the value of z = 20
the value of e = 3
the value of d = 7
Encrypted message is : 12.0
Decrypted message is : 12
```

## 9)DIFFE-HELL MAN:-

## Code:-

```
function power(a, b, p)
{
if (b == 1)
return a;
else
return((Math.pow(a, b)) % p);
}
```

```
// Driver code
var P, G, x, a, y, b, ka, kb;

// Both the persons will be agreed upon the
// public keys G and P

// A prime number P is taken
P = 23;
document.write("The value of P:"+ P + "<br>");

// A primitive root for P, G is taken
G = 9;
document.write("The value of G:"+ G +"<br>");

// Alice will choose the private key a
// a is the chosen private key
a = 4;

document.write("The private key a for Alice:" +
a +"<br>");

// Gets the generated key
x = power(G, a, P);

// Bob will choose the private key b
// b is the chosen private key
b = 3;
document.write("The private key b for Bob:" +
b + "<br>");
```

```
// Gets the generated key
y = power(G, b, P);

// Generating the secret key after the exchange
// of keys
ka = power(y, a, P); // Secret key for Alice
kb = power(x, b, P); // Secret key for Bob

document.write("Secret key for the Alice is:" +
ka +"<br>");
document.write("Secret key for the Bob is:" +
kb +"<br>");
```

## output:-

```
The value of P:23<br>
The value of G:9<br>
The private key a for Alice:4<br>
The private key b for Bob:3<br>
Secret key for the Alice is:9<br>
Secret key for the Bob is:9<br>
```

## 10)SHA-1 ALGORITHM:-

## Code:-

```
import java.security.*;
public class SHA1
{
public static void main(String[] a)
{
try
{
MessageDigest md = MessageDigest.getInstance("SHA1");
System.out.println("Message digest object info: ");
```

```
System.out.println(" Algorithm = " +md.getAlgorithm());
System.out.println("Provider = " +md.getProvider());
System.out.println(" ToString = " +md.toString());
String input = ""; md.update(input.getBytes());
byte[] output = md.digest(); System.out.println();
System.out.println("SHA1(\""+input+"\") = "+bytesToHex(output));
input = "abc"; md.update(input.getBytes());
output = md.digest(); System.out.println();
System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "abcdefghijklmnopqrstuvwxyz";
md.update(input.getBytes());

output = md.digest();
System.out.println();
System.out.println("SHA1(\"" +input+"\") = " +bytesToHex(output));
System.out.println("");

}
catch (Exception e)
{
System.out.println("Exception:" +e);
}
}
public static String bytesToHex(byte[] b)
{
char hexDigit[] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
StringBuffer buf = new StringBuffer();
for (int j=0; j<b.length; j++)
{
buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
buf.append(hexDigit[b[j] & 0x0f]);
```

```
}
return buf.toString();

}
}
```

## OUTPUT:-

Message digest object info:

Algorithm = SHA1

Provider = SUN version 19

ToString = SHA1 Message Digest from SUN, <initialized>

SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz") =
32D10C7B8CF96570CA04CE37F2A19D84240D3A89