Write a Java program that contains a string (char pointer) with a value 'Hello World'. The program should XOR each character in this string with 0 and displays the result.

```
public class XORWithZero
{
   public static void main(String[] args)
   {
      String text = "Hello World";
      System.out.println("Original String: " + text);
      System.out.print("XOR with 0: ");
      for (int i = 0; i < text.length(); i++)
      {
         char c = text.charAt(i);
         char xorResult = (char)(c ^ 0);
         System.out.print(xorResult);
      }
      System.out.println();
   }
}
```

Write a java program that contains a string (char pointer) with a value 'Hello World'. The program should AND or and XOR each character in this string with 127 and displays the result.

```
public class BitwiseOperations
{
   public static void main(String[] args)
   {
      String text = "Hello World";
      System.out.println("Original String: " + text);
      System.out.print("AND with 127: ");
      for (int i = 0; i < text.length(); i++)
      {
         char c = text.charAt(i);
         char andResult = (char)(c & 127);
         System.out.print(andResult);
      }
      System.out.println();
      System.out.print("XOR with 127: ");
      for (int i = 0; i < text.length(); i++)
      {
         char c = text.charAt(i);
         char xorResult = (char)(c ^ 127);
         System.out.print(xorResult);
      }
      System.out.println();
   }
}
```

Ceaser Cipher

```
import java.util.Scanner;
public class CaesarCipher {
   public static String encrypt(String message, int shift) {
      StringBuilder result = new StringBuilder();
      for (int i = 0; i < message.length(); i++) {
         char ch = message.charAt(i);
         if (Character.isUpperCase(ch)) {
            char c = (char) (((int) ch + shift - 65) % 26 + 65);
```

```
            result.append(c);
        }
        else if (Character.isLowerCase(ch)) {
            char c = (char) (((int) ch + shift - 97) % 26 + 97);
            result.append(c);
        }
        else {
            result.append(ch);
        }
    }
    return result.toString();
  }
  public static String decrypt(String message, int shift) {
    return encrypt(message, 26 - shift);
  }
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the message: ");
    String message = scanner.nextLine();
    System.out.print("Enter the shift value (1-25): ");
    int shift = scanner.nextInt();
    if (shift < 1 || shift > 25) {
      System.out.println("Invalid shift value. Please enter a number between 1 and 25.");
      return;
    }
    String encryptedMessage = encrypt(message, shift);
    System.out.println("Encrypted Message: " + encryptedMessage);
    String decryptedMessage = decrypt(encryptedMessage, shift);
    System.out.println("Decrypted Message: " + decryptedMessage);
    scanner.close();
  }
}
```

SubstitutionCipher

```
import java.util.Scanner;
public class SubstitutionCipher {
  private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  public static String encrypt(String message, String key) {
    StringBuilder encryptedMessage = new StringBuilder();
    message = message.toUpperCase();
    for (int i = 0; i < message.length(); i++) {
      char currentChar = message.charAt(i);
      if (Character.isLetter(currentChar)) {
        int indexInAlphabet = ALPHABET.indexOf(currentChar);
        char encryptedChar = key.charAt(indexInAlphabet);
        encryptedMessage.append(encryptedChar);
      } else {
        encryptedMessage.append(currentChar);
      }
    }
    return encryptedMessage.toString();
  }
  public static String decrypt(String encryptedMessage, String key) {
    StringBuilder decryptedMessage = new StringBuilder();
```

```
        encryptedMessage = encryptedMessage.toUpperCase();
        for (int i = 0; i < encryptedMessage.length(); i++) {
            char currentChar = encryptedMessage.charAt(i);
            if (Character.isLetter(currentChar)) {
                int indexInKey = key.indexOf(currentChar);
                char decryptedChar = ALPHABET.charAt(indexInKey);
                decryptedMessage.append(decryptedChar);
            } else {
                decryptedMessage.append(currentChar);
            }
        }
        return decryptedMessage.toString();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String key = "QWERTYUIOPLKJHGFDSAZXCVBNM";
        System.out.println("Using substitution key: " + key);
        System.out.print("Enter the message to encrypt: ");
        String message = scanner.nextLine();
        String encryptedMessage = encrypt(message, key);
        System.out.println("Encrypted Message: " + encryptedMessage);
        String decryptedMessage = decrypt(encryptedMessage, key);
        System.out.println("Decrypted Message: " + decryptedMessage);
        scanner.close();
    }
}
```

HillCipher

```
import java.util.Scanner;
public class HillCipher {
    public static int[] matrixMultiply(int[][] keyMatrix, int[] messageVector) {
        int[] result = new int[messageVector.length];
        for (int i = 0; i < keyMatrix.length; i++) {
            result[i] = 0;
            for (int j = 0; j < keyMatrix[i].length; j++) {
                result[i] += keyMatrix[i][j] * messageVector[j];
            }
            result[i] = result[i] % 26;
        }
        return result;
    }
    public static int modInverse(int a, int m) {
        a = a % m;
        for (int x = 1; x < m; x++) {
            if ((a * x) % m == 1) {
                return x;
            }
        }
        return 1;
    }
    public static int[][] inverseKeyMatrix(int[][] keyMatrix) {
        int determinant = (keyMatrix[0][0] * keyMatrix[1][1] - keyMatrix[0][1] * keyMatrix[1][0]) % 26;
        determinant = (determinant + 26) % 26;
        int inverseDeterminant = modInverse(determinant, 26);
```

```
        int[][] inverseMatrix = new int[2][2];
        inverseMatrix[0][0] = (keyMatrix[1][1] * inverseDeterminant) % 26;
        inverseMatrix[1][1] = (keyMatrix[0][0] * inverseDeterminant) % 26;
        inverseMatrix[0][1] = (-keyMatrix[0][1] * inverseDeterminant + 26) % 26;
        inverseMatrix[1][0] = (-keyMatrix[1][0] * inverseDeterminant + 26) % 26;
        return inverseMatrix;
    }
    public static int[] stringToVector(String text) {
        int[] vector = new int[text.length()];
        for (int i = 0; i < text.length(); i++) {
            vector[i] = text.charAt(i) - 'A';
        }
        return vector;
    }
    public static String vectorToString(int[] vector) {
        StringBuilder text = new StringBuilder();
        for (int i : vector) {
            text.append((char) (i + 'A'));
        }
        return text.toString();
    }
    public static String encrypt(String plaintext, int[][] keyMatrix) {
        int[] messageVector = stringToVector(plaintext);
        int[] encryptedVector = matrixMultiply(keyMatrix, messageVector);
        return vectorToString(encryptedVector);
    }
    public static String decrypt(String ciphertext, int[][] keyMatrix) {
        int[][] inverseMatrix = inverseKeyMatrix(keyMatrix);
        int[] messageVector = stringToVector(ciphertext);
        int[] decryptedVector = matrixMultiply(inverseMatrix, messageVector);
        return vectorToString(decryptedVector);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[][] keyMatrix = new int[2][2];
        System.out.println("Enter the 2x2 key matrix (values between 0 and 25):");
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                keyMatrix[i][j] = scanner.nextInt();
            }
        }
        System.out.println("Enter the plaintext (length 2, uppercase letters only):");
        String plaintext = scanner.next().toUpperCase();
        String ciphertext = encrypt(plaintext, keyMatrix);
        System.out.println("Encrypted Text: " + ciphertext);
        String decryptedText = decrypt(ciphertext, keyMatrix);
        System.out.println("Decrypted Text: " + decryptedText);
        scanner.close();
    }
}
```

Write a java program to implement the DES algorithm logic?
Write a java program to implement the Rijndael algorithm logic? ->AES
Write a java program to implement the Blowfish algorithm logic?

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
public class DESExample {
   public static SecretKey generateKey(int keySize) throws Exception {          ->for DES remove the parameter keySize
      KeyGenerator keyGenerator = KeyGenerator.getInstance("DES");
      keyGenerator.init(56);                          ->for other 2 except DES take keySize as function i/p and keep here
      return keyGenerator.generateKey();
   }
   public static String encrypt(String plaintext, SecretKey key) throws Exception {
      Cipher cipher = Cipher.getInstance("DES");
      cipher.init(Cipher.ENCRYPT_MODE, key);
      byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());
      return Base64.getEncoder().encodeToString(encryptedBytes);
   }
   public static String decrypt(String ciphertext, SecretKey key) throws Exception {
      Cipher cipher = Cipher.getInstance("DES");
      cipher.init(Cipher.DECRYPT_MODE, key);
      byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(ciphertext));
      return new String(decryptedBytes);
   }
   public static void main(String[] args) {
      try {
         SecretKey secretKey = generateKey();          ->for other 2 except DES pass 128 as arg to generateKey()
         String plaintext = "Hello, World!";
         System.out.println("Original Text: " + plaintext);
         String encryptedText = encrypt(plaintext, secretKey);
         System.out.println("Encrypted Text: " + encryptedText);
         String decryptedText = decrypt(encryptedText, secretKey);
         System.out.println("Decrypted Text: " + decryptedText);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Write a java program the RC4 logic using cryptography; encrypt the text "Hello World" using Blowfish. Create your own key using java key tool?

```
import java.util.Scanner;
public class RC4 {
   private byte[] S = new byte[256];
   private int x = 0;
   private int y = 0;
   public RC4(byte[] key) {
      init(key);
   }
   private void init(byte[] key) {
```

```
        int keyLength = key.length;
        for (int i = 0; i < 256; i++) {
            S[i] = (byte) i;
        }
        int j = 0;
        for (int i = 0; i < 256; i++) {
            j = (j + S[i] + key[i % keyLength]) & 0xFF;
            swap(i, j);
        }
    }
    private void swap(int i, int j) {
        byte temp = S[i];
        S[i] = S[j];
        S[j] = temp;
    }
    public byte[] encrypt(byte[] plaintext) {
        byte[] ciphertext = new byte[plaintext.length];
        for (int i = 0; i < plaintext.length; i++) {
            ciphertext[i] = (byte) (plaintext[i] ^ keyItem());
        }
        return ciphertext;
    }
    private byte keyItem() {
        x = (x + 1) & 0xFF;
        y = (y + S[x]) & 0xFF;
        swap(x, y);
        return S[(S[x] + S[y]) & 0xFF];
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a key for RC4 encryption (e.g., mysecretkey):");
        String keyString = scanner.nextLine();
        byte[] key = keyString.getBytes();
        RC4 rc4 = new RC4(key);
        String plaintext = "Hello World";
        System.out.println("Original Text: " + plaintext);
        byte[] ciphertext = rc4.encrypt(plaintext.getBytes());
        System.out.println("Encrypted Text: " + new String(ciphertext));
        byte[] decryptedText = rc4.encrypt(ciphertext);
        System.out.println("Decrypted Text: " + new String(decryptedText));
        scanner.close();
    }
}
```

Write a java program to implement RSA algorithm?

```
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.RSAPrivateKeySpec;
import java.security.spec.RSAPublicKeySpec;
```

```
import javax.crypto.Cipher;
public class RSAExample {
   public static void main(String[] args) {
      try {
         KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
         keyPairGenerator.initialize(2048);
         KeyPair keyPair = keyPairGenerator.generateKeyPair();
         PublicKey publicKey = keyPair.getPublic();
         PrivateKey privateKey = keyPair.getPrivate();
         printKeyDetails(publicKey, privateKey);
         String plaintext = "Hello, RSA!";
         System.out.println("Original Text: " + plaintext);
         byte[] encryptedText = encrypt(plaintext, publicKey);
         System.out.println("Encrypted Text: " + new String(encryptedText));
         String decryptedText = decrypt(encryptedText, privateKey);
         System.out.println("Decrypted Text: " + decryptedText);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   public static byte[] encrypt(String plaintext, PublicKey publicKey) throws Exception {
      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.ENCRYPT_MODE, publicKey);
      return cipher.doFinal(plaintext.getBytes());
   }
   public static String decrypt(byte[] ciphertext, PrivateKey privateKey) throws Exception {
      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.DECRYPT_MODE, privateKey);
      byte[] decryptedBytes = cipher.doFinal(ciphertext);
      return new String(decryptedBytes);
   }
   public static void printKeyDetails(PublicKey publicKey, PrivateKey privateKey) throws Exception {
      KeyFactory keyFactory = KeyFactory.getInstance("RSA");
      RSAPublicKeySpec publicKeySpec = keyFactory.getKeySpec(publicKey, RSAPublicKeySpec.class);
      RSAPrivateKeySpec privateKeySpec = keyFactory.getKeySpec(privateKey, RSAPrivateKeySpec.class);
      System.out.println("Public Key Modulus: " + publicKeySpec.getModulus());
      System.out.println("Public Key Exponent: " + publicKeySpec.getPublicExponent());
      System.out.println("Private Key Modulus: " + privateKeySpec.getModulus());
      System.out.println("Private Key Exponent: " + privateKeySpec.getPrivateExponent());
   }
}
```

Write a java program to calculate the message digest of text using the SHA-1 algorithm?

Write a java program to calculate the message digest of text using the MD5 algorithm?

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class SHA1DigestExample {
   public static void main(String[] args) {
      String input = "Hello, World!";
      try {
         MessageDigest md = MessageDigest.getInstance("SHA-1");          ->Instead of SHA-1 keep it as MD5 for other
         md.update(input.getBytes());
         byte[] digest = md.digest();
         StringBuilder sb = new StringBuilder();
```

```
            for (byte b : digest) {
                sb.append(String.format("%02x", b));
            }
            System.out.println("SHA-1 Digest: " + sb.toString());
        } catch (NoSuchAlgorithmException e) {
            System.out.println("SHA-1 algorithm not found: " + e.getMessage());
        }
    }
}
```