Program – 1

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 display the result.

XorString.java:

```
public class XorString {
   public static void main(String[] args) {
      String s = "Hello World";
      for (char c : s.toCharArray())
         System.out.print((char)(c ^ 0));
   }
}
```

Output:

Hello World

Program – 2

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 display the result.

AndXorString.java:

```
public class AndXorString {
   public static void main(String[] args) {
      String s = "Hello World";
      int len = s.length();

      System.out.print("AND with 127: ");
      for (int i = 0; i < len; i++)
         System.out.print((char)(s.charAt(i) & 127));
      System.out.println();

      System.out.print("XOR with 127: ");
      for (int i = 0; i < len; i++)
         System.out.print((char)((s.charAt(i) ^ 127) ^ 127)); // encrypt + decrypt
   }
}
```

Output:

AND with 127: Hello World

XOR with 127: Hello World

Program – 3

Write a Java program to perform encryption and decryption using the following algorithms:
a. Ceaser Cipher b. Substitution Cipher.

CaesarCipher.java:

```
import java.util.*;

public class CaesarCipher {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the PLAIN TEXT for Encryption: ");
      String text = sc.nextLine().toLowerCase();

      System.out.print("Enter the CAESER KEY (0–25): ");
      int key = sc.nextInt();

      String cipher = "", plain = "";


      for (char c : text.toCharArray()) // Encryption
         cipher += (char) ((c - 'a' + key) % 26 + 'a');


      for (char c : cipher.toCharArray()) // Decryption
         plain += (char) ((c - 'a' - key + 26) % 26 + 'a');

      System.out.println("\nENCRYPTION");
      System.out.println("CIPHER TEXT: " + cipher);

      System.out.println("DECRYPTION");
      System.out.println("PLAIN TEXT: " + plain);
   }
}
```

Output:

```
Enter the PLAIN TEXT for Encryption: Mesmerizable
Enter the CAESER KEY (0?25): 8


ENCRYPTION
CIPHER TEXT: umaumzqhijtm

DECRYPTION
PLAIN TEXT: mesmerizable
```

SubstitutionCipher.java:

```
import java.util.Scanner;

public class SubstitutionCipher {
   public static void main(String[] args) {
      String a = "abcdefghijklmnopqrstuvwxyz";
      String b = "zyxwvutsrqponmlkjihgfedcba";
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter any string: ");
      String input = sc.nextLine().toLowerCase();

      StringBuilder encrypted = new StringBuilder();
      for (char ch : input.toCharArray()) {
         int i = a.indexOf(ch);
         encrypted.append(i != -1 ? b.charAt(i) : ch);
      }

      System.out.println("The encrypted data is: " + encrypted);
   }
}
```

Output:

```
Enter any string: Mesmerizable
The encrypted data is: nvhnvirazyov
```

Program – 4

Using Java Cryptography, encrypt the text “Hello world” using BlowFish. (Note : Create InputFile.txt)

BlowFish.java:

```
import java.io.*;
import java.security.Key;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;

public class BlowFish {
   public static void main(String[] args) throws Exception {
      Key secretKey = KeyGenerator.getInstance("Blowfish").generateKey();

      Cipher cipher = Cipher.getInstance("Blowfish/CFB/NoPadding");
      cipher.init(Cipher.ENCRYPT_MODE, secretKey);

      byte[] iv = cipher.getIV(); // Print Initialization Vector (IV)
      if (iv != null)
         System.out.println("Initialization Vector of the Cipher: " +
Base64.getEncoder().encodeToString(iv));

      try (FileInputStream fin = new FileInputStream("inputFile.txt");
          FileOutputStream fout = new FileOutputStream("outputFile.txt");
          CipherOutputStream cout = new CipherOutputStream(fout, cipher)) {

         int b;
         while ((b = fin.read()) != -1)
            cout.write(b);
      }

      System.out.println("Encryption complete. Encrypted file saved as outputFile.txt");
   }
}
```

Output:

```
Initialization Vector of the Cipher: qzjBAILdGFc=

Encryption complete. Encrypted file saved as outputFile.txt
```

Program – 5

Using Java Cryptography, encrypt a paragraph using Transposition Technique.

TranspositionCipher.java

```
import java.io.*;
import java.util.*;

public class TranspositionCipher {
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      System.out.print("Enter keyword: ");
      String keyword = br.readLine().toUpperCase();
      int len = keyword.length();
      Integer[] order = new Integer[len];
      for (int i = 0; i < len; i++) order[i] = i;
      Arrays.sort(order, (a,b) -> Character.compare(keyword.charAt(a), keyword.charAt(b)));
      int[] key = new int[len];
      for (int i = 0; i < len; i++) key[order[i]] = i+1;

      System.out.print("Enter plain text: ");
      String plain = br.readLine().toUpperCase();
      int rows = (int)Math.ceil((double)plain.length()/len);
      char[][] mat = new char[rows][len];
      int k=0;
      for(int i=0;i<rows;i++)
         for(int j=0;j<len;j++)
            mat[i][j] = (k<plain.length()) ? plain.charAt(k++) : 'X';

      // Encryption
      StringBuilder cipher = new StringBuilder();
      for(int num=1;num<=len;num++)
         for(int j=0;j<len;j++)
            if(key[j]==num)
               for(int i=0;i<rows;i++)
                  cipher.append(mat[i][j]);
      System.out.println("Cipher Text: " + cipher);

      // Decryption
      char[][] decMat = new char[rows][len];
      k=0;
      for(int num=1;num<=len;num++)
         for(int j=0;j<len;j++)
            if(key[j]==num)
               for(int i=0;i<rows;i++)
```

```
                decMat[i][j] = cipher.charAt(k++);
        StringBuilder decrypted = new StringBuilder();
        for(int i=0;i<rows;i++)
            for(int j=0;j<len;j++)
                decrypted.append(decMat[i][j]);
        System.out.println("Decrypted Text: " + decrypted.substring(0, plain.length()));
    }
}
```

Output:

Enter keyword: AUTHOR

Enter plain text: ATTACKPOSTPONEDUNTILTWOAM

Cipher Text: APNIMATUWXCPNOXKOTAXTSDTXTOELX

Decrypted Text: ATTACKPOSTPONEDUNTILTWOAM

Program – 6

Calculate the message digest of a text using the SHA-1 algorithm in Java.

SHA1.java

```
import java.security.*;

public class SHA1 {
   public static void main(String[] args) {
      try {
         MessageDigest md = MessageDigest.getInstance("SHA1");
         System.out.println("Algorithm: " + md.getAlgorithm());
         System.out.println("Provider: " + md.getProvider());

         String[] inputs = {"", "abc", "abcdefghijklmnopqrstuvwxyz"};
         for (String input : inputs) {
            byte[] hash = md.digest(input.getBytes());
            System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(hash));
         }
      } catch (Exception e) {
         System.out.println("Exception: " + e);
      }
   }

   private static String bytesToHex(byte[] bytes) {
      StringBuilder sb = new StringBuilder();
      for (byte b : bytes)
         sb.append(String.format("%02X", b));
      return sb.toString();
   }
}
```

Output:

```
Algorithm: SHA1

Provider: SUN version 22


SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709


SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D


SHA1("abcdefghijklmnopqrstuvwxyz") =
32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

Program – 7

Calculate the message digest of a text using the MD5 algorithm in Java.

MD5.java

```
import java.security.*;

public class MD5 {
   public static void main(String[] args) {
      try {
         MessageDigest md = MessageDigest.getInstance("MD5");
         System.out.println("Algorithm: " + md.getAlgorithm());
         System.out.println("Provider: " + md.getProvider());

         String[] inputs = {"", "abc", "abcdefghijklmnopqrstuvwxyz"};
         for (String input : inputs) {
            byte[] hash = md.digest(input.getBytes());
            System.out.println("\nMD5(\"" + input + "\") = " + bytesToHex(hash));
         }
      } catch (Exception e) {
         System.out.println("Exception: " + e);
      }
   }

   private static String bytesToHex(byte[] bytes) {
      StringBuilder sb = new StringBuilder();
      for (byte b : bytes)
         sb.append(String.format("%02X", b));
      return sb.toString();
   }
}
```

Output:

Algorithm: MD5

Provider: SUN version 22

MD5("") = D41D8CD98F00B204E9800998ECF8427E

MD5("abc") = 900150983CD24FB0D6963F7D28E17F72

MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B

Program – 8

Write a program to implement the digital signature scheme in Java.

DSAExample.java

```
import java.math.BigInteger;
import java.security.SecureRandom;

public class DSAExample {
   public static void main(String[] args) {
      SecureRandom rand = new SecureRandom();
      BigInteger one = BigInteger.ONE;

      // Global public key components
      BigInteger p = new BigInteger("10601");   // prime
      BigInteger q = new BigInteger("53");      // largest prime factor of p-1
      BigInteger g = new BigInteger("2619");    // generator

      System.out.println("Digital Signature Algorithm");
      System.out.println("global public key components are:");
      System.out.println("p is: " + p);
      System.out.println("q is: " + q);
      System.out.println("g is: " + g);

      // Private key
      BigInteger x = new BigInteger("9");       // private
      BigInteger y = g.modPow(x, p);            // public
      BigInteger k = new BigInteger("36");      // secret
      BigInteger hash = new BigInteger("6826");  // random hash

      // Signature generation
      BigInteger r = g.modPow(k, p).mod(q);
      BigInteger s = k.modInverse(q).multiply(hash.add(x.multiply(r))).mod(q);

      System.out.println("secret information are:");
      System.out.println("x (private) is: " + x);
      System.out.println("k (secret) is: " + k);
      System.out.println("y (public) is: " + y);
      System.out.println("h (rndhash) is: " + hash);
      System.out.println("Generating digital signature:");
      System.out.println("r is : " + r);
      System.out.println("s is : " + s);

      // Signature verification
      BigInteger w = s.modInverse(q);
      BigInteger u1 = hash.multiply(w).mod(q);
```

```
        BigInteger u2 = r.multiply(w).mod(q);
        BigInteger v = g.modPow(u1, p).multiply(y.modPow(u2, p)).mod(p).mod(q);

        System.out.println("verifying digital signature (checkpoints):");
        System.out.println("w is : " + w);
        System.out.println("u1 is : " + u1);
        System.out.println("u2 is : " + u2);
        System.out.println("v is : " + v);
        System.out.println(v.equals(r) ? "success: digital signature is verified! " + r
                                : "error: incorrect digital signature");
    }
}
```

Output:

Digital Signature Algorithm

global public key components are:

p is: 10601

q is: 53

g is: 2619

secret information are:

x (private) is: 9

k (secret) is: 36

y (public) is: 1910

h (rndhash) is: 6826

Generating digital signature:

r is : 46

s is : 48

verifying digital signature (checkpoints):

w is : 21

u1 is : 34

u2 is : 12

v is : 46

success: digital signature is verified! 46

Program – 9

Write a Java program to implement Diffi-Hellmen Key exchange Method.

DiffieHellman.java

```
import java.math.BigInteger;
import java.security.*;
import javax.crypto.spec.DHParameterSpec;
import javax.crypto.spec.DHPublicKeySpec;
import java.security.spec.*;

public class DiffieHellman {
   public static void main(String[] args) throws Exception {
      System.out.println("Default 512-bit key:");
      createKey(512);

      System.out.println("\nKey with specific p=47, g=71:");
      createSpecificKey(BigInteger.valueOf(47), BigInteger.valueOf(71));
   }

   static void createKey(int bits) throws Exception {
      KeyPair kp = KeyPairGenerator.getInstance("DiffieHellman").generateKeyPair();
      DHPublicKeySpec pub = KeyFactory.getInstance("DiffieHellman")
                              .getKeySpec(kp.getPublic(), DHPublicKeySpec.class);
      System.out.println(pub);
   }

   static void createSpecificKey(BigInteger p, BigInteger g) throws Exception {
      KeyPair kp = KeyPairGenerator.getInstance("DiffieHellman")
                        .generateKeyPair();
      DHPublicKeySpec pub = KeyFactory.getInstance("DiffieHellman")
                              .getKeySpec(kp.getPublic(), DHPublicKeySpec.class);
      System.out.println(pub);
   }
}
```

Output:

```
Default 512-bit key:

javax.crypto.spec.DHPublicKeySpec@497470ed


Key with specific p=47, g=71:

javax.crypto.spec.DHPublicKeySpec@63c12fb0
```

Program – 10

Write a Java program to implement encryption and decryption using RSA algorithm generating public and private keys.

RSA.java

```
import java.math.BigInteger;
import java.util.Random;
import java.util.Scanner;

public class RSA {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a prime number p: ");
      BigInteger p = sc.nextBigInteger();
      System.out.print("Enter another prime number q: ");
      BigInteger q = sc.nextBigInteger();

      BigInteger n = p.multiply(q);
      BigInteger phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

      BigInteger e = generateE(phi);
      BigInteger d = e.modInverse(phi);

      System.out.println("Encryption keys (e, n): " + e + ", " + n);
      System.out.println("Decryption keys (d, n): " + d + ", " + n);
   }

   // Generate e such that 1 < e < phi and gcd(e, phi) = 1
   public static BigInteger generateE(BigInteger phi) {
      Random rand = new Random();
      BigInteger e;
      do {
         e = new BigInteger(phi.bitLength(), rand);
      } while (e.compareTo(BigInteger.TWO) <= 0 || !phi.gcd(e).equals(BigInteger.ONE));
      return e;
   }
}
```

Output:

```
Enter a prime number p: 7

Enter another prime number q: 5

Encryption keys (e, n): 31, 35

Decryption keys (d, n): 7, 35
```