Practical 1:-

/*

//monoaplhabeticcipher:-

```
package com.mycompany.caesarcipher;



import java.util.Scanner;

public class CaesarCipher {

   public static char p[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
'v', 'w', 'x', 'y', 'z'};
   public static char ch[] = {'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Z',
'X', 'C', 'V', 'B', 'N', 'M'};

   public static String doEncryption(String s) {
      char c[] = new char[(s.length())];
      for (int i = 0; i < s.length(); i++) {
         for (int j = 0; j < 26; j++) {
            if (p[j] == s.charAt(i)) {
               c[i] = ch[j];
               break;
            }
         }
      }
      return (new String(c));
   }

   public static String doDecryption(String s) {
```

```
        char p1[] = new char[(s.length())];
        for (int i = 0; i < s.length(); i++) {
            for (int j = 0; j < 26; j++) {
                if (ch[j] == s.charAt(i)) {
                    p1[i] = p[j];
                    break;
                }
            }
        }
        return (new String(p1));
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the message : ");
        String en = doEncryption(sc.next().toLowerCase());
        System.out.println("Encrypted message : " + en);
        System.out.println("Decrypted message : " + doDecryption(en));
        sc.close();
//      --comehometoday
    }
}

*/

//--------------------------------------------------------------------------------------------------------------




/*
//Modified Caesar Cipher:-
```

```
package com.mycompany.caesarcipher;

import java.util.Scanner;

public class CaesarCipher {

    public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

    public static 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;
    }

    public static 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;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the key : ");
        int shiftKey = sc.nextInt();
        System.out.println("Enter the string for encryption: ");
        String message = new String();
        message = sc.next();
        System.out.println(encrypt(message, shiftKey));
        System.out.println(decrypt(encrypt(message, shiftKey), shiftKey));
        sc.close();
//      --4
//      --come
    }
}
*/

//-----------------------------------------------------------------------------------------------------------



/*
//Caesar Cipher:-

package com.mycompany.caesarcipher;

import java.util.Scanner;
```

```
public class CaesarCipher {

   public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

   public static 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;
   }

   public static 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;
   }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string for encryption: ");
        String message = new String();
        message = sc.next();
        System.out.println(encrypt(message, 3));
        System.out.println(decrypt(encrypt(message, 3), 3));
        sc.close();
//      --meetmeafterthetogoparty
    }
}*/
```

Practial 2:-

/*

//monoaplhabeticcipher:-

```
package com.mycompany.caesarcipher;



import java.util.Scanner;

public class CaesarCipher {

   public static char p[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
'v', 'w', 'x', 'y', 'z'};
   public static char ch[] = {'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Z',
'X', 'C', 'V', 'B', 'N', 'M'};

   public static String doEncryption(String s) {
      char c[] = new char[(s.length())];
      for (int i = 0; i < s.length(); i++) {
         for (int j = 0; j < 26; j++) {
            if (p[j] == s.charAt(i)) {
               c[i] = ch[j];
               break;
            }
         }
      }
      return (new String(c));
   }

   public static String doDecryption(String s) {
```

```
        char p1[] = new char[(s.length())];
        for (int i = 0; i < s.length(); i++) {
            for (int j = 0; j < 26; j++) {
                if (ch[j] == s.charAt(i)) {
                    p1[i] = p[j];
                    break;
                }
            }
        }
        return (new String(p1));
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the message : ");
        String en = doEncryption(sc.next().toLowerCase());
        System.out.println("Encrypted message : " + en);
        System.out.println("Decrypted message : " + doDecryption(en));
        sc.close();
//      --comehometoday
    }
}

*/

//------------------------------------------------------------------------------------------------------------



/*
//Modified Caesar Cipher:-
```

```
package com.mycompany.caesarcipher;

import java.util.Scanner;

public class CaesarCipher {

    public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

    public static 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;
    }

    public static 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;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the key : ");
        int shiftKey = sc.nextInt();
        System.out.println("Enter the string for encryption: ");
        String message = new String();
        message = sc.next();
        System.out.println(encrypt(message, shiftKey));
        System.out.println(decrypt(encrypt(message, shiftKey), shiftKey));
        sc.close();
//      --4
//      --come
    }
}
*/

//-----------------------------------------------------------------------------------------------------------



/*
//Caesar Cipher:-

package com.mycompany.caesarcipher;

import java.util.Scanner;
```

```
public class CaesarCipher {

   public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

   public static 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;
   }

   public static 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;
   }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string for encryption: ");
        String message = new String();
        message = sc.next();
        System.out.println(encrypt(message, 3));
        System.out.println(decrypt(encrypt(message, 3), 3));
        sc.close();
//      --meetmeafterthetogoparty
    }
}*/
```

Practical 3:-

/*

//simplecolumnarcipher:-

```
package com.mycompany.railfence;
import java.util.*;

public class Simplecolumnarcipher {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter plaintext : ");
      String message = sc.nextLine();
      System.out.println("Enter key in number : ");
      String key = sc.nextLine();
      int columnCount = key.length();
      int rowCount = (message.length() + columnCount - 1) / columnCount;
      int plainText[][] = new int[rowCount][columnCount];
      int cipherText[][] = new int[rowCount][columnCount];
      System.out.print("\n Encryption");
      cipherText = encrypt(plainText, cipherText, message, rowCount, columnCount, key
      );
      String ct = "";
      for (int i = 0; i < columnCount; i++) {
         for (int j = 0; j < rowCount; j++) {
            if (cipherText[j][i] == 0) {
               ct = ct + 'x';
            } else {
               ct = ct + (char) cipherText[j][i];
            }
```

```
        }
      }
      System.out.print("\n Cipher Text : " + ct.toString());
      System.out.print("\n Decryption");
      plainText = decrypt(plainText, cipherText, ct, rowCount, columnCount, key);
      String pt = "";
      for (int i = 0; i < rowCount; i++) {
        for (int j = 0; j < columnCount; j++) {
          if (plainText[i][j] == 0) {
            pt = pt + "";
          } else {
            pt = pt + (char) plainText[i][j];
          }
        }
      }
      System.out.print("Plain text : " + pt);
      System.out.println();
    }

    static int[][] encrypt(int plainText[][], int cipherText[][], String message, int rowCount, int
columnCount, String key) {
      int i, j;
      int k = 0;
      for (i = 0; i < rowCount; i++) {
        for (j = 0; j < columnCount; j++) {
          if (k < message.length()) {
            plainText[i][j] = (int) message.charAt(k);
            k++;
          } else {
            plainText[i][j] = 'x';
          }
```

```
        }
    }
    for (i = 0; i < columnCount; i++) {
        int currentCol = ((int) key.charAt(i) - 48) - 1;
        for (j = 0; j < rowCount; j++) {
            cipherText[j][i] = plainText[j][currentCol];
        }
    }
    System.out.print("Cipher array \n");
    for (i = 0; i < rowCount; i++) {
        for (j = 0; j < columnCount; j++) {
            System.out.print((char) cipherText[i][j] + "");
        }
        System.out.println();
    }
    return cipherText;
}

static int[][] decrypt(int plainText[][], int cipherText[][], String message, int rowCount, int
columnCount, String key) {
    int i, j;
    for (i = 0; i < columnCount; i++) {
        int currentCol = ((int) key.charAt(i) - 48) - 1;
        for (j = 0; j < rowCount; j++) {
            plainText[j][currentCol] = cipherText[j][i];
        }
    }
    System.out.print("Plain array \n");
    for (i = 0; i < rowCount; i++) {
        for (j = 0; j < columnCount; j++) {
            System.out.print((char) plainText[i][j] + "\t");
```

```
            }
            System.out.println();
        }
        return plainText;
//      --attackpostponeduntiltwoam
//      --4312567
    }
}



*/

//-------------------------------------------------------------------------------------------------------------------------
------------



/*

//vernamecipher:-



package com.mycompany.railfence;
import java.util.Scanner;

public class VernamCipher {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter String : ");
        String txt = sc.nextLine();
        System.out.println("Enter OTP(One-Time Pad): ");
```

```
        String otp = sc.nextLine();
        String st = "";
        char m, n;
        int p1 = 0, p2 = 2;
        char c[] = new char[]{'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z'};
        int n1[] = new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25};
        if (txt.length() != otp.length()) {
            System.out.println("Please enter OTP as the same length of string : ");
            otp = sc.nextLine();
        }
        for (int i = 0; i < txt.length(); i++) {
            m = (char) (txt.charAt(i));
            n = (char) (otp.charAt(i));
            for (int j = 0; j < c.length; j++) {
                if (m == c[j]) {
                    p1 = n1[j];
                }
                if (n == c[j]) {
                    p2 = n1[j];
                }
            }
            int p = p1 + p2;
            System.out.println(p1 + "+" + p2 + "=");
            System.out.println(p);
            if (p >= 26) {
                p = p - 26;
            }
            char c1 = c[p];
            System.out.println("\n\tCHARACTER at " + p + " is " + c1);
            st = st + c1;
```

```
        }
        System.out.println("_________________");
        System.out.println("Cipher text is : " + st);
//      --howareyou
//      --ncbtzqarx
    }
}

*/

//-----------------------------------------------------------------------------------------------------------------------

/*

//Rail Fence Cipher :-

package com.mycompany.railfence;

public class Railfence {

    public static void main(String[] args) {
        String input = "meetmeafterthetogaparty";
        String output = "";
        int len = input.length();
        int flag = 0;
        System.out.println("Input string : " + input);
        for (int i = 0; i < len; i += 2) {
            output += input.charAt(i);
        }
        for (int i = 1; i < len; i += 2) {
            output += input.charAt(i);
```

```
        }
        System.out.println("Ciphered Text : " + output);
    }
}


*/
```

Practical 4:-

```
/*

//DES Algorithm:-
//Part1:-

package com.mycompany.destest1;

import javax.crypto.Cipher;
import javax.crypto.SecretKeyFactory;
import javax.crypto.SecretKey;
import javax.crypto.spec.DESKeySpec;
import java.util.BASE64Encoder;
import java.util.BASE64Dncoder;

public class Destest1 {

   private SecretKey key;
   public String theKey;

   public void generateKey() throws Exception {
      DESKeySpec deskeySpec = new DESKeySpec(theKey.getBytes());
      SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
      key = keyFactory.generateSecret(deskeySpec);
   }

   public String encrypt(String messg) throws Exception {
      Cipher cipher = Cipher.getInstance("DES");
      cipher.init(cipher.ENCRYPT_MODE, key);
      byte[] stringBytes = messg.getBytes("UTF-8");
```

```
    byte[] raw = cipher.doFinal(stringBytes);
    BASE64Encoder encode = new BASE64Encoder();
    String base64 = encode.encode(raw);
    return base64;
}

public String decrypt(String encrypted) throws Exception {
    Cipher cipher = Cipher.getInstance("DES");
    cipher.init(cipher.DECRYPT_MODE, key);
    BASE64Decoder decode = new BASE64Decoder();
    byte[] raw = decode.decodeBuffer(encrypted);
    byte[] stringBytes = cipher.doFinal(raw);
    String clear = new String(stringBytes, "UTF-8");
    return clear;
}

public static void main(String[] args) {
    String messg = "Shallun Monteiro";
    String decrypted;
    String encrypted;
    Destest1 des = new Destest1();
    des.theKey = "1,2,3,4,5,6";
    try {
        des.generateKey();
        System.out.println("Clear Message: " + messg);
        encrypted = des.encrypt(messg);
        decrypted = des.decrypt(encrypted);
        System.out.println("Encrypted Message: " + encrypted);
        System.out.println("Decrypted Message: " + decrypted);
    } catch (Exception e) {
    }
```

```
    }
}

*/

//-----------------------------------------------------------------------------------------------------------------



/*
//DES Algorithm:-
//Part2:-

package com.mycompany.destest1;

import javax.crypto.Cipher;
import javax.crypto.SecretKeyFactory;
import javax.crypto.SecretKey;
import javax.crypto.spec.DESKeySpec;
import java.util.Base64;

public class Destest1 {

    private SecretKey key;
    public String theKey;

    public void generateKey() throws Exception {
        DESKeySpec deskeySpec = new DESKeySpec(theKey.getBytes());
        SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
        key = keyFactory.generateSecret(deskeySpec);
    }
```

```
public String encrypt(String messg) throws Exception {
   Cipher cipher = Cipher.getInstance("DES");
   cipher.init(cipher.ENCRYPT_MODE, key);
   byte[] stringBytes = messg.getBytes("UTF-8");
   byte[] raw = cipher.doFinal(stringBytes);
   String base64 = Base64.getEncoder().encodeToString(raw);
   return base64;
}

public String decrypt(String encrypted) throws Exception {
   Cipher cipher = Cipher.getInstance("DES");
   cipher.init(cipher.DECRYPT_MODE, key);
   byte[] raw = Base64.getDecoder().decode(encrypted);
   byte[] stringBytes = cipher.doFinal(raw);
   String clear = new String(stringBytes, "UTF-8");
   return clear;
}

public static void main(String[] args) {
   String messg = "Shallun Monteiro";
   String decrypted;
   String encrypted;
   Destest1 des = new Destest1();
   des.theKey = "12345678"; // Note: DES key should be 8 bytes long
   try {
      des.generateKey();
      System.out.println("Clear Message: " + messg);
      encrypted = des.encrypt(messg);
      decrypted = des.decrypt(encrypted);
      System.out.println("Encrypted Message: " + encrypted);
      System.out.println("Decrypted Message: " + decrypted);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

*/

//-----------------------------------------------------------------------------------------------------------------------



/*
//AES Algorithm:-
//Part1:-

package com.mycompany.destest1;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.security.Key;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class Aestest {

    private byte[] keyValue;

    public Aestest(String key) {
        keyValue = key.getBytes();
    }
```

```
private Key generateKey() throws Exception {
    Key key = new SecretKeySpec(keyValue, "AES");
    return key;
}

public String encrypt(String messg) throws Exception {
    Key key = generateKey();
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] raw = cipher.doFinal(messg.getBytes());
    BASE64Encoder encoder = new BASE64Encoder();
    String base64 = encoder.encode(raw);
    return base64;
}

public String decrypt(String encrypted) throws Exception {
    Key key = generateKey();
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, key);
    BASE64Decoder decoder = new BASE64Decoder();
    byte[] raw = decoder.decodeBuffer(encrypted);
    byte[] stringBytes = cipher.doFinal(raw);
    String clear = new String(stringBytes, "UTF8");
    return clear;
}

public static void main(String[] args) {
    String messg = "MITTU DON";
    String decrypted;
    String encrypted;
    Aestest aest = new Aestest("1v39eptlvuhaqqsr");
```

```
        try {
            System.out.println("AES:");
            System.out.println("Clear Message: " + messg);
            encrypted = aest.encrypt(messg);
            System.out.println("Encrypted Message: " + encrypted);
            decrypted = aest.decrypt(encrypted);
            System.out.println("Decrypted Message: " + decrypted);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

*/

//-----------------------------------------------------------------------------------------------------------------------

/*

//AES Algorithm:-
//Part2:-

package com.mycompany.destest1;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.security.Key;
import java.util.Base64;

public class Aestest {
```

```
private byte[] keyValue;

public Aestest(String key) {
   keyValue = key.getBytes();
}

private Key generateKey() throws Exception {
   Key key = new SecretKeySpec(keyValue, "AES");
   return key;
}

public String encrypt(String messg) throws Exception {
   Key key = generateKey();
   Cipher cipher = Cipher.getInstance("AES");
   cipher.init(Cipher.ENCRYPT_MODE, key);
   byte[] raw = cipher.doFinal(messg.getBytes());
   String base64 = Base64.getEncoder().encodeToString(raw);
   return base64;
}

public String decrypt(String encrypted) throws Exception {
   Key key = generateKey();
   Cipher cipher = Cipher.getInstance("AES");
   cipher.init(Cipher.DECRYPT_MODE, key);
   byte[] raw = Base64.getDecoder().decode(encrypted);
   byte[] stringBytes = cipher.doFinal(raw);
   String clear = new String(stringBytes, "UTF-8");
   return clear;
}

public static void main(String[] args) {
```

```
        String messg = "MITTU DON";
        String decrypted;
        String encrypted;
        Aestest aest = new Aestest("1v39eptlvuhaqqsr");
        try {
           System.out.println("AES:");
           System.out.println("Clear Message: " + messg);
           encrypted = aest.encrypt(messg);
           System.out.println("Encrypted Message: " + encrypted);
           decrypted = aest.decrypt(encrypted);
           System.out.println("Decrypted Message: " + decrypted);
        } catch (Exception e) {
           e.printStackTrace();
        }
     }
}

*/
```

Practical 5:-

```
package com.mycompany.rsaalgorithm;

import java.math.*;
import java.util.*;

public class RSAalgorithm {

    public static void main(String[] args) {
        int p, q, n, phi, d = 0, e, i;
        int msg = 10;
        double c;
        BigInteger msgback;
        p = 7;
        q = 17;
        n = p * q;
        phi = (p - 1) * (q - 1);
        System.out.println("The value of z = " + phi);
        for (e = 2; e < phi; e++) {
            if (gcd(e, phi) == 1) {
                break;
            }
        }
        System.out.println("The value of e = " + e);
        for (i = 0; i <= 9; i++) {
            int x = 1 + (i * phi);
            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);
        BigInteger N = BigInteger.valueOf(n);
        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);
        }
    }
}
```

Practical 6:-

```
package com.mycompany.diffiehellman;

public class DiffieHellman {

   private static long power(long a, long b, long p) {
      if (b == 1) {
         return a;
      } else {
         return (((long) Math.pow(a, b)) % p);
      }
   }

   public static void main(String[] args) {
      long n, g, x, A, y, B, Ka, Kb;
      n = 11;
      System.out.println("The value of N : " + n);
      g = 7;
      System.out.println("The value of g : " + g);
      x = 3;
      System.out.println("The private key for Alice : " + x);
      A = power(g, x, n);
      System.out.println("Value of A --> " + A);
      y = 6;
      System.out.println("The private key for Bob : " + y);
      B = power(g, y, n);
      System.out.println("Value of B --> " + B);
      Ka = power(B, x, n);
      Kb = power(A, y, n);
      System.out.println("Secret key for Alice is : " + Ka);
```

```
        System.out.println("Secret key for Bob is : " + Kb);
    }
}
```

Practical 7:-

```
package com.mycompany.mdhash;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MDHash {

    public static void main(String[] args) {
        System.out.println("MD Algorithm");
        System.out.println("For null " + md5(""));
        System.out.println("For Simple text" + md5("This is my text"));
        System.out.println("For Simple numbers" + md5("12345"));
    }

    public static String md5(String input) {
        String md5 = null;
        if (null == input) {
            return null;
        }
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5");
            digest.update(input.getBytes(), 0, input.length());
            md5 = new BigInteger(1, digest.digest()).toString(16);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return md5;
    }
```

```
}
```

Practical 8:-

```
package com.mycompany.hmac;

import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class HMAC {

   static public byte[] calcHmacSha256(byte[] secretKey, byte[] message) {
      byte[] hmacSha256 = null;
      try {
         Mac mac = Mac.getInstance("HmacSHA256");
         SecretKeySpec secretKeySpec = new SecretKeySpec(secretKey, "HmacSHA256");
         mac.init(secretKeySpec);
         hmacSha256 = mac.doFinal(message);
      } catch (Exception e) {
         throw new RuntimeException("Failed to calculate hmac-sha256", e);
      }
      return hmacSha256;
   }

   public static void main(String[] args) {
      try {
         byte[] hmacSha256;
```

```
            hmacSha256 = HMAC.calcHmacSha256("secret123".getBytes("UTF-8"), "hello
world".getBytes("UTF-8"));

            System.out.println("Implementing SHA algorithm");

            System.out.println(String.format("Hex: %032x", new BigInteger(1, hmacSha256)));

        } catch (UnsupportedEncodingException e) {

            e.printStackTrace();

        }

    }

}
```

Practical 9:-

/*\

//SSLServer:-

```
package com.mycompany.sslclient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.ssl.SSLServerSocketFactory;

public class SSLServer {

   static final int port = 8000;

   public static void main(String[] args) {
      SSLServerSocketFactory sslServerSocketFactory = (SSLServerSocketFactory)
SSLServerSocketFactory.
            getDefault();
      try {
         ServerSocket sslServerSocket = sslServerSocketFactory.createServerSocket(port);
         System.out.println("SSL ServerSocket Started");
         System.out.println(sslServerSocket.toString());
         Socket socket = sslServerSocket.accept();
```

```
        System.out.println("ServerSocket Accepted");
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        try (BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()))) {
            String line;
            while ((line = bufferedReader.readLine()) != null) {
               System.out.println(line);
               out.println(line);
            }
         }
         System.out.println("Closed");
      } catch (IOException ex) {
         Logger.getLogger(SSLServer.class.getName()).log(Level.SEVERE, null, ex);
      }
   }
}

 */


//-------------------------------------------------------------------------------------------------------------------


/*

//SSLClient:-

package com.mycompany.sslclient;

import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSocketFactory;

public class SSLClient {

   static final int port = 8000;

   public static void main(String[] args) {
      SSLSocketFactory sslSocketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
      try {
         Socket socket = sslSocketFactory.createSocket("localhost", port);
         PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
         try (BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()))) {
            Scanner scanner = new Scanner(System.in);
            while (true) {
               System.out.println("Enter something ");
               String inputLine = scanner.nextLine();
               if (inputLine.equals("q")) {
                  break;
               }
               out.println(inputLine);
               System.out.println(bufferedReader.readLine());
            }
         }
```

```
        } catch (IOException ex) {
            Logger.getLogger(SSLClient.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}


*/
```