TASK 1

Task 1(a):Write a Java program to perform encryption and decryption using the algorithm Ceaser Cipher.

```
import java.util.*;
public class Main{
   public static final String alpha="abcdefghijklmnopqrstuvwxyz";
   public static String enc(String inp,int key) throws Exception{
      String encrypt="";
      inp = inp.toLowerCase();
      for(int i=0;i<inp.length();i++){
         int index=alpha.indexOf(inp.charAt(i));
         if(index!=-1){
         index = (index+key)%26;
         encrypt+=alpha.charAt(index);
         }
        else{
            encrypt+=inp.charAt(i);
        }
      }
      return encrypt;
   }
   public static String dec(String inp,int key) throws Exception{
      String decrypt="";
      inp = inp.toLowerCase();
      for(int i=0;i<inp.length();i++){
         int index=alpha.indexOf(inp.charAt(i));
         index = index-key;
```

```
        if(index<0){
          index=index+26;
          decrypt+=alpha.charAt(index);
        }
        else{
          decrypt+=alpha.charAt(index);
        }

      }
      return decrypt;
    }
    public static void main(String[] args) throws Exception{
      Scanner sc = new Scanner(System.in);
      String inp = sc.nextLine();
      String en = enc(inp,3);
      System.out.println(en);
      String de = dec(en,3);
      System.out.println(de);
    }
}
```

Task 1(b): Write a Java program to perform encryption and decryption using the algorithm Substitution Cipher.

```
import java.util.*;
class Main {
  public static final String a = "abcdefghijklmnopqrstuvwxyz";
  public static final String b = "zyxwvutsrqponmlkjihgfedcba";

  // Encryption method
  public static String enc(String inp) {
    String encrypt = "";
    for (int i = 0; i < inp.length(); i++) {
      int index = a.indexOf(inp.charAt(i));
      if (index < 0) {
        encrypt += inp.charAt(i); // Keeps non-alphabetic characters unchanged
      } else {
        encrypt += b.charAt(index); // Maps alphabetic characters to their reverse
      }
    }
    return encrypt;
  }

  // Decryption method
  public static String dec(String inp) {
    String decrypt = "";
    for (int i = 0; i < inp.length(); i++) {
      int index = b.indexOf(inp.charAt(i));
      if (index < 0) {
        decrypt += inp.charAt(i); // Keeps non-alphabetic characters unchanged
```

```
        } else {
            decrypt += a.charAt(index); // Maps reverse characters back to original
        }
      }
      return decrypt;
    }

    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the string to encrypt:");
      String input = sc.nextLine();

      // Encrypt the input
      String encrypted = enc(input.toLowerCase());
      System.out.println("Encrypted string: " + encrypted);

      // Decrypt the encrypted string
      String decrypted = dec(encrypted);
      System.out.println("Decrypted string: " + decrypted);
    }
}
```

TASK 2:

Implement symmetric block cipher encryption and decryption using DES algorithm in C/JAVA.

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
```

```
class Main{
public static SecretKey gen() throws Exception{
KeyGenerator kg = KeyGenerator.getInstance("DES");
kg.init(56);
return kg.generateKey();
}
public static String enc(String inp,SecretKey key) throws Exception{
Cipher cipher= Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE,key);
byte[] b=cipher.doFinal(inp.getBytes());
return Base64.getEncoder().encodeToString(b);
}
public static String dec(String ctext,SecretKey sk) throws Exception{
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.DECRYPT_MODE,sk);
byte[] b1 = Base64.getDecoder().decode(ctext);
byte[] b2 = cipher.doFinal(b1);
return new String(b2);
}
public static void main (String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
SecretKey sk = gen();
System.out.println("Enter a text");
String pt = sc.nextLine();
String et = enc(pt,sk);
System.out.println("The encrypted text is ");
System.out.println(et);
System.out.println("The decrypted text is");
```

```
String dt = dec(et,sk);
System.out.println(dt);
}
}
```

TASK 3:

Write a C/JAVA program to implement encryption technique using Blowfish algorithm.

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
class Main{
public static SecretKey gen() throws Exception{
KeyGenerator kg = KeyGenerator.getInstance("Blowfish");
kg.init(128);
return kg.generateKey();
}
public static String enc(String inp,SecretKey key) throws Exception{
Cipher cipher= Cipher.getInstance("Blowfish");
cipher.init(Cipher.ENCRYPT_MODE,key);
byte[] b=cipher.doFinal(inp.getBytes());
return Base64.getEncoder().encodeToString(b);
}
public static String dec(String ctext,SecretKey sk) throws Exception{
Cipher cipher = Cipher.getInstance("Blowfish");
cipher.init(Cipher.DECRYPT_MODE,sk);
byte[] b1 = Base64.getDecoder().decode(ctext);
byte[] b2 = cipher.doFinal(b1);
```

```
return new String(b2);
}
public static void main (String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
SecretKey sk = gen();
System.out.println("Enter a text");
String pt = sc.nextLine();
String et = enc(pt,sk);
System.out.println("The encrypted text is ");
System.out.println(et);
System.out.println("The decrypted text is");
String dt = dec(et,sk);
System.out.println(dt);
}
}
```

TASK 4:

Implement the encryption of block chunk of 128 bits size using AES algorithm in C/JAVA.

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
```

```
class Main{
public static SecretKey gen() throws Exception{
KeyGenerator kg = KeyGenerator.getInstance("AES");
kg.init(128);
return kg.generateKey();
}
public static String enc(String inp,SecretKey key) throws Exception{
Cipher cipher= Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE,key);
byte[] b=cipher.doFinal(inp.getBytes());
return Base64.getEncoder().encodeToString(b);
}
public static String dec(String ctext,SecretKey sk) throws Exception{
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.DECRYPT_MODE,sk);
byte[] b1 = Base64.getDecoder().decode(ctext);
byte[] b2 = cipher.doFinal(b1);
return new String(b2);
}
public static void main (String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
SecretKey sk = gen();
System.out.println("Enter a text");
String pt = sc.nextLine();
String et = enc(pt,sk);
System.out.println("The encrypted text is ");
System.out.println(et);
System.out.println("The decrypted text is");
```

```
String dt = dec(et,sk);
System.out.println(dt);
}
}
```

TASK 5:

Write a C/JAVA program on Rivest Cipher 4(RC4) logic.

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
class Main {
```

```
public static SecretKey gen() throws Exception {
KeyGenerator kg = KeyGenerator.getInstance("RC4");
kg.init(128);
return kg.generateKey();
}
public static String enc(String inp,SecretKey key) throws Exception {
Cipher cipher= Cipher.getInstance("RC4");
cipher.init(Cipher.ENCRYPT_MODE,key);
byte[] b=cipher.doFinal(inp.getBytes());
return Base64.getEncoder().encodeToString(b);
}
public static String dec(String ctext,SecretKey sk) throws Exception {
Cipher cipher = Cipher.getInstance("RC4");
cipher.init(Cipher.DECRYPT_MODE,sk);
byte[] b1 = Base64.getDecoder().decode(ctext);
byte[] b2 = cipher.doFinal(b1);
return new String(b2);
}
public static void main (String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
SecretKey sk = gen();
System.out.println("Enter a text");
String pt = sc.nextLine();
String et = enc(pt,sk);
System.out.println("The encrypted text is ");
System.out.println(et);
System.out.println("The decrypted text is");
String dt = dec(et,sk);
```

```
System.out.println(dt);
}
}
```

TASK 6:

Implement DES-2 and DES-3 using Java cryptography package.

```
#DES-2
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
```

```
class Main {
public static SecretKey gen() throws Exception {
KeyGenerator kg = KeyGenerator.getInstance("DES");
kg.init(56);
return kg.generateKey();
}
public static String enc(String inp,SecretKey key1,SecretKey key2) throws Exception
{
Cipher cipher= Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE,key1);
byte[] first=cipher.doFinal(inp.getBytes());
cipher.init(Cipher.ENCRYPT_MODE,key2);
byte[] second= cipher.doFinal(first);
return Base64.getEncoder().encodeToString(second);
}
public static String dec(String ctext,SecretKey key1,SecretKey key2) throws
Exception {
Cipher cipher = Cipher.getInstance("DES");
byte[] b1 = Base64.getDecoder().decode(ctext);
cipher.init(Cipher.DECRYPT_MODE,key2);
byte[] initial = cipher.doFinal(b1);
cipher.init(Cipher.DECRYPT_MODE,key1);
byte[] original = cipher.doFinal(initial);
return new String(original);
}
public static void main (String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
SecretKey sk1 = gen();
```

```
SecretKey sk2 = gen();
System.out.println("Enter a text");
String pt = sc.nextLine();
String et = enc(pt,sk1,sk2);
System.out.println("The encrypted text is ");
System.out.println(et);
System.out.println("The decrypted text is");
String dt = dec(et,sk1,sk2);
System.out.println(dt);
}
}
```

#DES-3

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
class Main {
public static SecretKey gen() throws Exception {
KeyGenerator kg = KeyGenerator.getInstance("DESede");
kg.init(168);
return kg.generateKey();
}
public static String enc(String inp,SecretKey key) throws Exception {
Cipher cipher= Cipher.getInstance("DESede");
cipher.init(Cipher.ENCRYPT_MODE,key);
byte[] b=cipher.doFinal(inp.getBytes());
return Base64.getEncoder().encodeToString(b);
}
```

```
public static String dec(String ctext,SecretKey sk) throws Exception {
Cipher cipher = Cipher.getInstance("DESede");
cipher.init(Cipher.DECRYPT_MODE,sk);
byte[] b1 = Base64.getDecoder().decode(ctext);
byte[] b2 = cipher.doFinal(b1);
return new String(b2);
}
public static void main (String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
SecretKey sk = gen();
System.out.println("Enter a text");
String pt = sc.nextLine();
String et = enc(pt,sk);
System.out.println("The encrypted text is ");
System.out.println(et);
System.out.println("The decrypted text is");
String dt = dec(et,sk);
System.out.println(dt);
}
}
```

TASK 7:

Design a Java program to implement RSA algorithm.

```
import java.security.*;
import javax.crypto.Cipher;
import java.util.Base64;
public class Main {
// Generate RSA Key Pair
```

```
public static KeyPair generateKeyPair() throws Exception {
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
keyPairGen.initialize(2048); // 2048-bit key size
return keyPairGen.generateKeyPair();
}
// Encrypt message using public key
public static String encrypt(String message, PublicKey publicKey) throws Exception {
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.ENCRYPT_MODE, publicKey);
byte[] encrypted = cipher.doFinal(message.getBytes());
return Base64.getEncoder().encodeToString(encrypted); // Convert encrypted bytes to
}
// Decrypt message using private key
public static String decrypt(String encryptedMessage, PrivateKey privateKey) throws
Exception {
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, privateKey);
byte[] decrypted = cipher.doFinal(Base64.getDecoder().decode(encryptedMessage)); //
return new String(decrypted);
}
public static void main(String[] args) throws Exception {
// Generate RSA Key Pair
KeyPair keyPair = generateKeyPair();
PublicKey publicKey = keyPair.getPublic();
PrivateKey privateKey = keyPair.getPrivate();
// Message to encrypt
String originalMessage = "Hello, RSA!";
System.out.println("Original Message: " + originalMessage);
```

```
// Encrypt the message
String encryptedMessage = encrypt(originalMessage, publicKey);
System.out.println("Encrypted Message: " + encryptedMessage);
// Decrypt the message
String decryptedMessage = decrypt(encryptedMessage, privateKey);
System.out.println("Decrypted Message: " + decryptedMessage);
}
}
```

TASK 8:

Implement key exchange protocol using the Diffie-Hellman algorithm.

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.math.BigInteger;
import java.security.*;
class Main{
public static final BigInteger p = new BigInteger("23");
public static final BigInteger g = new BigInteger("5");
public static BigInteger gen(){
SecureRandom random = new SecureRandom();
return new BigInteger(256,random);
}
private static BigInteger calpub(BigInteger pk) {
return g.modPow(pk, p);
}
```

```
private static BigInteger calsh(BigInteger opuk, BigInteger prk) {
return opuk.modPow(prk, p); // Shared secret = otherPublicKey^privateKey mod p
}
public static void main (String[] args) {
BigInteger aliceprivate = gen();
BigInteger bobprivate = gen();
BigInteger alicepublic = calpub(aliceprivate);
BigInteger bobpublic = calpub(bobprivate);
BigInteger aliceshared = calsh(bobpublic,aliceprivate);
BigInteger bobshared = calsh(alicepublic,bobprivate);
System.out.println(aliceshared);
System.out.println(bobshared);
}
}
```

TASK 9:

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

```
import java.security.MessageDigest;
import java.util.Base64;
import java.util.Scanner;
public class Main {
public static String sha1Hash(String input) throws Exception {
```

```
// Initialize SHA-1 MessageDigest
MessageDigest md = MessageDigest.getInstance("SHA-1");
// Compute hash
byte[] hashBytes = md.digest(input.getBytes());
// Convert byte array to a Base64-encoded string for readability
return Base64.getEncoder().encodeToString(hashBytes);
}
public static void main(String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
System.out.println("Enter a String:");
String input = sc.nextLine();
// Hash input and display
String hashedOutput = sha1Hash(input);
System.out.println("SHA-1 Hash: " + hashedOutput);
}
}
```

TASK 10:

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

```
import java.security.*;
import java.util.*;
public class Main {
public static String sha1Hash(String input) throws Exception {
// Initialize SHA-1 MessageDigest
```

```
MessageDigest md = MessageDigest.getInstance("md5");
// Compute hash
byte[] hashBytes = md.digest(input.getBytes());
// Convert byte array to a Base64-encoded string for readability
return Base64.getEncoder().encodeToString(hashBytes);
}
public static void main(String[] args) throws Exception {
Scanner sc = new Scanner(System.in);
System.out.println("Enter a String:");
String input = sc.nextLine();
// Hash input and display
String hashedOutput = sha1Hash(input);
System.out.println("MD5 Hash: " + hashedOutput);
}
}
```

Task12:

Write a program in java, which performs a digital signature on a given text.

```
import java.security.*;
import java.util.*;
public class Main {
public static void main(String[] args) throws Exception {
// Sample text to sign
```

```
Scanner sc = new Scanner(System.in);
String inp= sc.nextLine();
// Generate a key pair (private and public keys)
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
keyGen.initialize(2048);
KeyPair keyPair = keyGen.generateKeyPair();
PrivateKey privateKey = keyPair.getPrivate();
PublicKey publicKey = keyPair.getPublic();
// Sign the data
Signature signature = Signature.getInstance("SHA256withRSA");
signature.initSign(privateKey);
signature.update(inp.getBytes());
byte[] digitalSignature = signature.sign();
// Encode the signature in Base64 for easier display
String encodedSignature = Base64.getEncoder().encodeToString(digitalSignature);
System.out.println("Digital Signature: " + encodedSignature);
// Verify the signature
signature.initVerify(publicKey);
signature.update(inp.getBytes());
boolean isVerified = signature.verify(digitalSignature);
// Output the verification result
System.out.println("Is the signature verified? " + isVerified);
}
}
```