Practical No. 4

Aim:- To write a program to implement DES algorithm.

Theory :- DES stands for Data Encryption standard. There are certain machines that can be used to create the DES algorithm. The DES algorithm uses a key of 56-bit size. Using this key. The DES takes a block of 64bit plain text as input and generate a block of 64-bit cipher text.

Initial Permutation (IP) =>
The plain text is divided into smaller chunks of 64-bit size. The IP is performed before the first round. for eg. the 58th bit replaces the first bit and so on. The resultant 64 bit is split into two equal halves of 32 bit each called left plain text (LPT) and right plain text(RPT

Step 1: Key Transformation => We already know that the DES process uses a 56-bit key which is obtained by eliminating all the bits present in every 8th position in a 64-bit key. In this step, a 48-bit key is generated.

Step 2: Expansion Permutation => The RPT of 32 bit size is broken down into 8 chunks of 4 bits each and extra two bits are added to every chunk leading to 48-bit data.

Algorithm

The steps for this algorithm are as follows:-

1> The process begins with 64 bit plain text handed over to IP function.
2> The IP is then performed on the plain text.
3> Next, the IP creates two halves referred to as LPT and RPT.
4> Each LPT and RPT goes through 16 rounds of the encryption process.

5> Finally ,the LPT and RPT are rejoined and a final Permutation (FP) is performed on the newly combined block.

6> The result of this process produced the desired 64-bit cipher text.

Conclusion :- DES for encryption and decryption completed successfully.

Program:

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;

public class DES {

  public static byte[] encrypt(String keyStr, String plaintext) throws Exception {
    byte[] keyBytes = hexStringToByteArray(keyStr);
    SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "DES");

    Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec);

    return cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));
  }

  public static String decrypt(String keyStr, byte[] ciphertext) throws Exception {
    byte[] keyBytes = hexStringToByteArray(keyStr);
    SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "DES");

    Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, keySpec);

    byte[] decryptedBytes = cipher.doFinal(ciphertext);
    return new String(decryptedBytes, StandardCharsets.UTF_8);
  }

  public static byte[] hexStringToByteArray(String hex) {
    int len = hex.length();
    byte[] data = new byte[len / 2];
    for (int i = 0; i < len; i += 2) {
      data[i / 2] = (byte) ((Character.digit(hex.charAt(i), 16) << 4)
          + Character.digit(hex.charAt(i + 1), 16));
    }
    return data;
  }

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

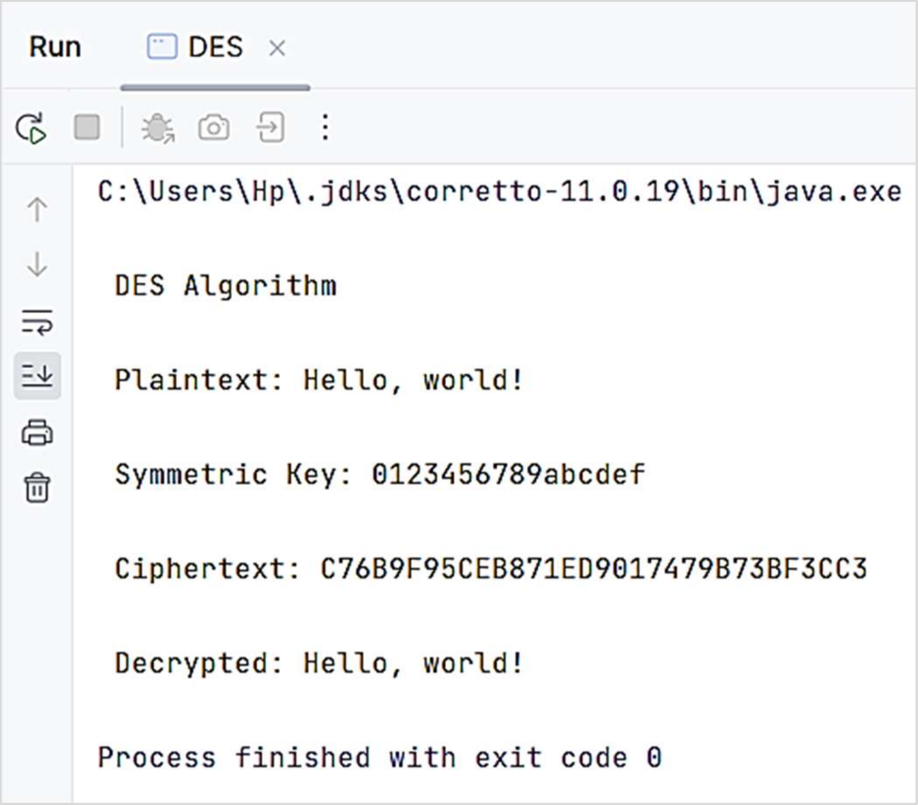
```
public static void main(String[] args) throws Exception {
    // Define DES key and plaintext
    String key = "0123456789abcdef";
    String plaintext = "Hello, world!";

    // Perform DES encryption
    byte[] ciphertext = encrypt(key, plaintext);

    // Perform DES decryption
    String decrypted = decrypt(key, ciphertext);

    // Print results
    System.out.println("\n DES Algorithm");
    System.out.println("\n Plaintext: " + plaintext);
    System.out.println("\n Symmetric Key: "+ key);
    System.out.println("\n Ciphertext: " + bytesToHex(ciphertext));
    System.out.println("\n Decrypted: " + decrypted);
  }
```

Output:



```
C:\Users\Hp\.jdks\corretto-11.0.19\bin\java.exe

 DES Algorithm

 Plaintext: Hello, world!

 Symmetric Key: 0123456789abcdef

 Ciphertext: C76B9F95CEB871ED9017479B73BF3CC3

 Decrypted: Hello, world!

Process finished with exit code 0
```