PRACTICAL - 6

AIM:

Simplified DES (S-DES) is a symmetric-key block cipher. The S-DES encryption algorithm takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8-bit block of ciphertext as output. It follows two rounds. Implement S-DES symmetric encryption Algorithm.

THEORY:

Simplified Data Encryption Standard is a simple version of Data Encryption Standard having a 10-bit key and 8-bit plain text. It is much smaller than the DES algorithm as it takes only 8-bit plain text whereas DES takes 64-bit plain text. It was developed for educational purpose so that understanding DES can become easy. It is a block cipher algorithm and uses a symmetric key for its algorithm i.e. they use the same key for both encryption and decryption. It has 2 rounds for encryption which use two different keys.

CODE:

```
public class Main {
    // int key[]= {0,0,1,0,0,1,0,1,1,1};
    int key[] = {
            1, 0, 1, 0, 0, 0, 0, 0, 1, 0
    }; // extra example for checking purpose
    int P10[] = { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6 };
    int P8[] = { 6, 3, 7, 4, 8, 5, 10, 9 };

    int key1[] = new int[8];
    int key2[] = new int[8];

    int[] IP = { 2, 6, 3, 1, 4, 8, 5, 7 };
    int[] EP = { 4, 1, 2, 3, 2, 3, 4, 1 };
    int[] P4 = { 2, 4, 3, 1 };
    int[] IP_inv = { 4, 1, 3, 5, 7, 2, 8, 6 };

    int[][] S0 = { { 1, 0, 3, 2 },
            { 3, 2, 1, 0 },
            { 0, 2, 1, 3 },
            { 3, 1, 3, 2 } };
    int[][] S1 = { { 0, 1, 2, 3 },
            { 2, 0, 1, 3 },
            { 3, 0, 1, 0 },
            { 2, 1, 0, 3 } };

    //    this function basically generates the key(key1 and
    //key2)     using P10 and P8 with (1 and 2)left shifts

    void key_generation()
    {
        int key_[] = new int[10];
```

```
        for (int i = 0; i < 10; i++) {
            key_[i] = key[P10[i] - 1];
        }

        int Ls[] = new int[5];
        int Rs[] = new int[5];

        for (int i = 0; i < 5; i++) {
            Ls[i] = key_[i];
            Rs[i] = key_[i + 5];
        }

        int[] Ls_1 = shift(Ls, 1);
        int[] Rs_1 = shift(Rs, 1);

        for (int i = 0; i < 5; i++) {
            key_[i] = Ls_1[i];
            key_[i + 5] = Rs_1[i];
        }

        for (int i = 0; i < 8; i++) {
            key1[i] = key_[P8[i] - 1];
        }

        int[] Ls_2 = shift(Ls, 2);
        int[] Rs_2 = shift(Rs, 2);

        for (int i = 0; i < 5; i++) {
            key_[i] = Ls_2[i];
            key_[i + 5] = Rs_2[i];
        }

        for (int i = 0; i < 8; i++) {
            key2[i] = key_[P8[i] - 1];
        }

        System.out.println("Your Key-1 :");

        for (int i = 0; i < 8; i++)
            System.out.print(key1[i] + " ");

        System.out.println();
        System.out.println("Your Key-2 :");

        for (int i = 0; i < 8; i++)
            System.out.print(key2[i] + " ");
    }

    //    this function is use full for shifting(circular) the
    //array n position towards left
```

```
int[] shift(int[] ar, int n)
{
   while (n > 0) {
      int temp = ar[0];
      for (int i = 0; i < ar.length - 1; i++) {
         ar[i] = ar[i + 1];
      }
      ar[ar.length - 1] = temp;
      n--;
   }
   return ar;
}

//    this is main encryption function takes plain text as
//input     uses another functions and returns the array of
//cipher text

int[] encryption(int[] plaintext)
{
   int[] arr = new int[8];

   for (int i = 0; i < 8; i++) {
      arr[i] = plaintext[IP[i] - 1];
   }
   int[] arr1 = function_(arr, key1);

   int[] after_swap = swap(arr1, arr1.length / 2);

   int[] arr2 = function_(after_swap, key2);

   int[] ciphertext = new int[8];

   for (int i = 0; i < 8; i++) {
      ciphertext[i] = arr2[IP_inv[i] - 1];
   }

   return ciphertext;
}

// decimal to binary string 0-3

String binary_(int val)
{
   if (val == 0)
      return "00";
   else if (val == 1)
      return "01";
   else if (val == 2)
      return "10";
   else
      return "11";
```

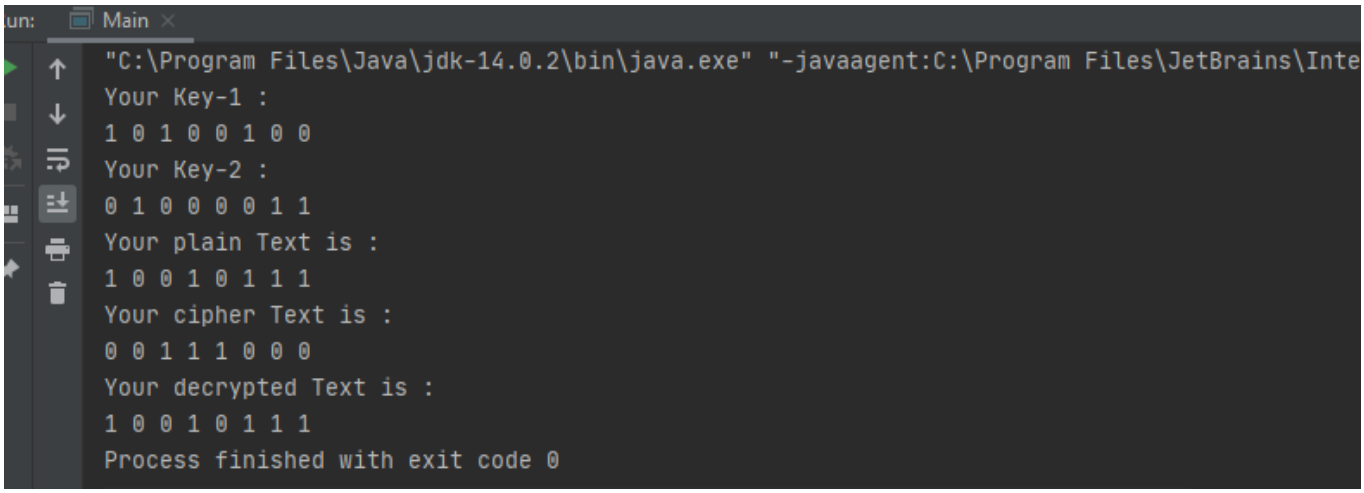
```
}

//    this function is doing core things like expansion
//    then xor with desired key then S0 and S1
//substitution     P4 permutation and again xor     we have used
//this function 2 times(key-1 and key-2) during
//encryption and     2 times(key-2 and key-1) during
//decryption

int[] function_(int[] ar, int[] key_)
{

   int[] l = new int[4];
   int[] r = new int[4];

   for (int i = 0; i < 4; i++) {
      l[i] = ar[i];
      r[i] = ar[i + 4];
   }

   int[] ep = new int[8];

   for (int i = 0; i < 8; i++) {
      ep[i] = r[EP[i] - 1];
   }

   for (int i = 0; i < 8; i++) {
      ar[i] = key_[i] ^ ep[i];
   }

   int[] l_1 = new int[4];
   int[] r_1 = new int[4];

   for (int i = 0; i < 4; i++) {
      l_1[i] = ar[i];
      r_1[i] = ar[i + 4];
   }

   int row, col, val;

   row = Integer.parseInt("" + l_1[0] + l_1[3], 2);
   col = Integer.parseInt("" + l_1[1] + l_1[2], 2);
   val = S0[row][col];
   String str_l = binary_(val);

   row = Integer.parseInt("" + r_1[0] + r_1[3], 2);
   col = Integer.parseInt("" + r_1[1] + r_1[2], 2);
   val = S1[row][col];
   String str_r = binary_(val);

   int[] r_ = new int[4];
```

```
        for (int i = 0; i < 2; i++) {
            char c1 = str_l.charAt(i);
            char c2 = str_r.charAt(i);
            r_[i] = Character.getNumericValue(c1);
            r_[i + 2] = Character.getNumericValue(c2);
        }
        int[] r_p4 = new int[4];
        for (int i = 0; i < 4; i++) {
            r_p4[i] = r_[P4[i] - 1];
        }

        for (int i = 0; i < 4; i++) {
            l[i] = l[i] ^ r_p4[i];
        }

        int[] output = new int[8];
        for (int i = 0; i < 4; i++) {
            output[i] = l[i];
            output[i + 4] = r[i];
        }
        return output;
    }

    //    this function swaps the nibble of size n(4)

    int[] swap(int[] array, int n)
    {
        int[] l = new int[n];
        int[] r = new int[n];

        for (int i = 0; i < n; i++) {
            l[i] = array[i];
            r[i] = array[i + n];
        }

        int[] output = new int[2 * n];
        for (int i = 0; i < n; i++) {
            output[i] = r[i];
            output[i + n] = l[i];
        }

        return output;
    }

    //    this is main decryption function
    //    here we have used all previously defined function
    //    it takes cipher text as input and returns the array
    //of     decrypted text

    int[] decryption(int[] ar)
    {
```

```
        int[] arr = new int[8];

        for (int i = 0; i < 8; i++) {
            arr[i] = ar[IP[i] - 1];
        }

        int[] arr1 = function_(arr, key2);

        int[] after_swap = swap(arr1, arr1.length / 2);

        int[] arr2 = function_(after_swap, key1);

        int[] decrypted = new int[8];

        for (int i = 0; i < 8; i++) {
            decrypted[i] = arr2[IP_inv[i] - 1];
        }

        return decrypted;
    }

    public static void main(String[] args)
    {

        Main obj = new Main();

        obj.key_generation(); // call to key generation
        // function

        // int []plaintext= {1,0,1,0,0,1,0,1};
        int[] plaintext = {
                1, 0, 0, 1, 0, 1, 1, 1
        }; // extra example for checking purpose

        System.out.println();
        System.out.println("Your plain Text is :");
        for (int i = 0; i < 8; i++) // printing the
            // plaintext
            System.out.print(plaintext[i] + " ");

        int[] ciphertext = obj.encryption(plaintext);

        System.out.println();
        System.out.println(
                "Your cipher Text is :"); // printing the cipher
        // text
        for (int i = 0; i < 8; i++)
            System.out.print(ciphertext[i] + " ");

        int[] decrypted = obj.decryption(ciphertext);
```

```
            System.out.println();
            System.out.println(
                  "Your decrypted Text is :"); // printing the
            // decrypted text
            for (int i = 0; i < 8; i++)
               System.out.print(decrypted[i] + " ");
         }
}
```

OUTPUT:



LATEST APPLICATIONS:

- DES is a public-key cryptosystem that is widely used for secure data transmission.

LEARNING OUTCOME:

➔ Through this practical I have learned how to generate key for DES Encryption

REFERENCES:

- https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/