Lab – 8,9

Subject : NIS

Aim :- Implement DES (Data Encryption Standard) Encryption and Decryption algorithm with Key Generation.

Program: -

```
expansion_box = [32, 1 , 2 , 3 , 4 , 5 , 4 , 5,
         6 , 7 , 8 , 9 , 8 , 9 , 10, 11,
         12, 13, 12, 13, 14, 15, 16, 17,
         16, 17, 18, 19, 20, 21, 20, 21,
         22, 23, 24, 25, 24, 25, 26, 27,
         28, 29, 28, 29, 30, 31, 32, 1 ]
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]
final_perm = [ 40, 8, 48, 16, 56, 24, 64, 32,
               39, 7, 47, 15, 55, 23, 63, 31,
               38, 6, 46, 14, 54, 22, 62, 30,
```

```
                37, 5, 45, 13, 53, 21, 61, 29,

                36, 4, 44, 12, 52, 20, 60, 28,

                35, 3, 43, 11, 51, 19, 59, 27,

                34, 2, 42, 10, 50, 18, 58, 26,

                33, 1, 41, 9, 49, 17, 57, 25]

sbox =  [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

          [ 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

          [ 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

          [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 ]],


         [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],

           [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],

           [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],

          [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 ]],


         [ [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],

           [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],

           [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],

            [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 ]],


          [ [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],

           [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],

           [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],

            [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14] ],
```

```
          [ [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
           [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
            [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
           [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 ]],


         [ [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
           [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
            [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
            [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13] ],


          [ [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
           [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
            [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
            [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12] ],


         [ [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
            [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
            [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
            [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11] ]
            ]
key_comp = [14, 17, 11, 24, 1, 5,
            3, 28, 15, 6, 21, 10,
            23, 19, 12, 4, 26, 8,
            16, 7, 27, 20, 13, 2,
            41, 52, 31, 37, 47, 55,
```

```
            30, 40, 51, 45, 33, 48,

            44, 49, 39, 56, 34, 53,

            46, 42, 50, 36, 29, 32]

per = [ 16,  7, 20, 21,

        29, 12, 28, 17,

         1, 15, 23, 26,

         5, 18, 31, 10,

         2,  8, 24, 14,

        32, 27,  3,  9,

        19, 13, 30,  6,

        22, 11, 4, 25]

keyp = [57, 49, 41, 33, 25, 17, 9,

        1, 58, 50, 42, 34, 26, 18,

        10, 2, 59, 51, 43, 35, 27,

        19, 11, 3, 60, 52, 44, 36,

        63, 55, 47, 39, 31, 23, 15,

        7, 62, 54, 46, 38, 30, 22,

        14, 6, 61, 53, 45, 37, 29,

        21, 13, 5, 28, 20, 12, 4 ]

def permute(k, arr, n):

    permutation = ""

    for i in range(0, n):

        permutation = permutation + k[arr[i] - 1]

    return permutation

key_list=[]
```

```
def key_generation(key):
  key=permute(key,keyp,56)
  l,r=key[:28],key[28:]
  for i in range(16):
    string=""
    if i == 0 or i == 1 or i == 8 or i == 15:
      l=l[1:28]+l[0]
      r=r[1:28]+r[0]
    else:
      l=l[2:28]+l[:2]
      r=r[2:28]+r[:2]
    joined=l+r
    string=permute(joined,key_comp,48)
    key_list.append(string)
def encrypt(ip):
  ip=permute(ip,initial_perm,64)
  left,right=ip[:32],ip[32:64]
  for i in range(16):
    xored=""
    r_output=round_fun(right,key_list[i])
    for j in range(32):
      xored+=str(int(left[j]) ^ int(r_output[j]))
    left=xored
    # print("#",i)
    # print("Left",left)
```

```
    # print("Right",right)
    if i!=15:
      left,right= right,left
  cipher_text=left+right
  cipher_text=permute(cipher_text,final_perm,64)
  return cipher_text
def decrypt(ip):
  return encrypt(ip)
def round_fun(ip,key):
  expansion_list=""
  xored=""
  #doing expansion
  expansion_list=permute(ip,expansion_box,48)
  #xoring
  for i in range(len(key)):
    xored+=str(int(expansion_list[i]) ^ int(key[i]))
  #print(len(xored))
  sbox_str=""
  #finding 32 output using s-boxes
  k=0
  for i in range(0,len(xored),6):
    #print(i)
    l_temp=xored[i:i+6]
    #print(len(l_temp))
    row=binary_to_decimal(l_temp[0]+l_temp[-1])
```

```
    column=binary_to_decimal(l_temp[1:5])
    sbox_str+=str(decimal_to_binary(sbox[k][row][column]))
    k+=1
  sbox_str=permute(sbox_str,per,32)
  return sbox_str
def decimal_to_binary(n):
    binary_arr=[]
    if n == 0:
        return '0000'
    while(n > 0):
        binary_arr.append(str(n%2))
        n=n//2
    binary_arr.reverse()
    while len(binary_arr) < 4:
      binary_arr.insert(0,'0')
    return "".join(binary_arr)
def binary_to_decimal(s):
  l=len(s)
  s=s[::-1]
  ans=0
  for i in range(l):
    ans+=(2**i)*int(s[i])
  return ans
if __name__=='__main__':
  bit_input=input()
```

```
key=input()

key_generation(key)

encrypted_text=encrypt(bit_input)

key_list=key_list[::-1]

decrypted_text=decrypt(encrypted_text)

print("Encrypted :",encrypted_text)

print("Decrypted :",decrypted_text)
```

Output: -

```
PS D:\DDIT CE\Sem 6\NIS\Lab 8> python .\DES.py
Enter 64 bit plaintext: 1010101010010101010101011011101001100101001010001110010110010101
Enter 64 bit key: 1011111100011001111100110001101011111101111000101001111100101010
Encrypted : 0110010101000000011100101110100111110101000110101010000101110000
Decrypted : 1010101010010101010101011011101001100101001010001110010110010101
PS D:\DDIT CE\Sem 6\NIS\Lab 8>
```