Student name: Edvinas Dulskas
Student code:
19040186

source code ***sdes.py***

```
import random



# --------- Functions -----------

# key generation function (9bit)

def sdes_genkey():



        key = ''

        for i in range(9):

                key += str(random.randint(0,1))



        return key



# function for making 8bit from 9bit key

def key_8bit(key, n):



        new_key = ''

        idx = n

        for i in range(8):

                if idx == len(key)+1:

                        idx = 1

                new_key += key[idx-1]

                idx += 1
```

Student name: Edvinas Dulskas
Student code:
19040186

```
        return new_key


# simple DES encryption function
def sdes_encrypt(key, pblock):

        ctx = pblock

        for i in range(3):
                # copy L and R sides (both 6bits)
                L = ctx[:6]
                R = ctx[6:]
                # regenerate key (8 bit)
                k2 = key_8bit(key, i+2)                 # i+2 = 2, 3, 4 (key row number)
                # go to function and expand R 6bit side to 8bit lenght message
                Rf = R[:2] + R[3] + R[2] + R[3] + R[2] + R[4:]
                # temp - for storing R(8 bit) side xor key(idx)
                temp = ''
                for i in range(len(Rf)):
                        temp += str(int(Rf[i]) ^ int(k2[i]))            # xor
operation
                # spliting answer into 2x4bits
                L_4bit = temp[:4]                       # used for 4bit left side in function
                R_4bit = temp[4:]                       # used for 4bit right side in function
                # taking values from SBoxes and joining them into 1 6bit string
```

```
                fresult = get_sbox_value(L_4bit, 1)
                fresult += get_sbox_value(R_4bit, 2)
                # function output xor L side
                tt = ''
                for i in range(len(fresult)):
                        tt += str(int(L[i]) ^ int(fresult[i]))
                # final ciphertext of the round
                # in this case R is new L and tt is new R
                ctx = R + tt

        return ctx

# simple DES decryption function
def sdes_decrypt(key, cblock):

        pp = cblock

        for i in range(3):
                # we know that Ln = Rn-1 (present L block is past R block)
                Rpast = pp[:6]
                # we need to make 8bit key again
                k2 = key_8bit(key, 4-i)
                # the steps are the same like in encryption, we need to get the function value
                # we need to make R side 8bit
```

Student name: Edvinas Dulskas
Student code:
19040186

```
                Rf = Rpast[:2] + Rpast[3] + Rpast[2] + Rpast[3] + Rpast[2] + Rpast[4:]

                # now we have to use xor for k2 and Rf

                temp = ''

                for i in range(len(Rf)):

                        temp += str(int(Rf[i]) ^ int(k2[i]))

                # spliting answer into 2x4bits

                L_4bit = temp[:4]                       # used for 4bit left side in function

                R_4bit = temp[4:]                       # used for 4bit right side in function

                # now we need to take values from SBoxes and merge them into 6bit message

                fresult = get_sbox_value(L_4bit, 1)

                fresult += get_sbox_value(R_4bit, 2)

                # now we need to take present R block

                Rpresent = pp[6:]

                # let's find the past L block by using xor (present R block xor function result)

                Lpast = ''

                for i in range(len(Rpresent)):

                        Lpast += str(int(Rpresent[i]) ^ int(fresult[i]))

                pp = Lpast + Rpast


        return pp


# function for getting SBoxes values

def get_sbox_value(value, box_idx):
```

Student name: Edvinas Dulskas
Student code:
19040186

```
result = ''                                                    # empty string for 3bit result

s1 = [['101', '010', '001', '110', '011', '100', '111', '000'],

                    ['001', '100', '110', '010', '000', '111', '101', '011']]

s2 = [['100', '000', '110', '101', '111', '001', '011', '010'],

                    ['101', '011', '000', '111', '110', '010', '001', '100']]



if box_idx == 1:

                    sbox = s1

else:

                    sbox = s2



# Sbox1

if value[0] == '0':

                    #take 1st row

                    if value[1:] == '000':              # col: 0

                                        result += sbox[0][0]

                    elif value[1:] == '001':            # col: 1

                                        result += sbox[0][1]

                    elif value[1:] == '010':            # col: 2

                                        result += sbox[0][2]

                    elif value[1:] == '011':            # col: 3

                                        result += sbox[0][3]

                    elif value[1:] == '100':            # col: 4

                                        result += sbox[0][4]
```

```
            elif value[1:] == '101':                # col: 5
                    result += sbox[0][5]
            elif value[1:] == '110':                # col: 6
                    result += sbox[0][6]
            elif value[1:] == '111':                # col: 7
                    result += sbox[0][7]
    else:
            #take 2nd row
            if value[1:] == '000':                  # col: 0
                    result += sbox[1][0]
            elif value[1:] == '001':                # col: 1
                    result += sbox[1][1]
            elif value[1:] == '010':                # col: 2
                    result += sbox[1][2]
            elif value[1:] == '011':                # col: 3
                    result += sbox[1][3]
            elif value[1:] == '100':                # col: 4
                    result += sbox[1][4]
            elif value[1:] == '101':                # col: 5
                    result += sbox[1][5]
            elif value[1:] == '110':                # col: 6
                    result += sbox[1][7]
            elif value[1:] == '111':                # col: 7
                    result += sbox[1][7]
```

Student name: Edvinas Dulskas
Student code:
19040186

```
                return result


# --------- Main code -----------

msg = '111100001100'


key = sdes_genkey()

nkey = key_8bit(key, 4)


C = sdes_encrypt(key, msg)

P = sdes_decrypt(key, C)


print('Key: ' + key + '\nOriginal message:  ' + msg + '\nEncrypted message: ' + C + '\nDecrypted message: ' + P )
```

*Results of **program**:*