Name: Abdurrahman Qureshi

Roll No: 242466

Practical No: 6

Date Of Performance: 04/08/2025

Aim: Implement DES technique to encrypt and decrypt message, Use java or python code

<u>CODE:</u>

```
from typing import List

IP = [
    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
]

FP = [
    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
```

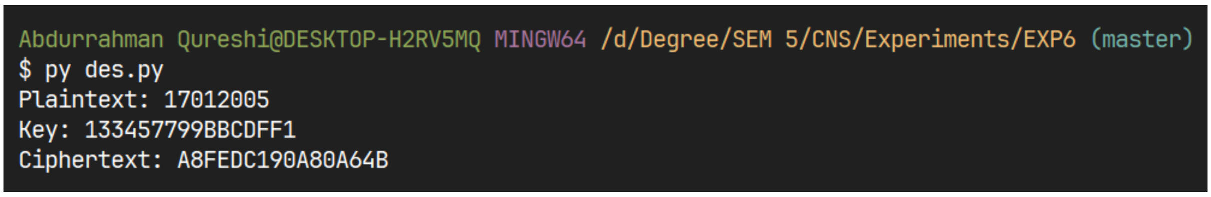
```
]

E = [
    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
]

S_BOX = [

    [
        [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]
    ]
]

P = [
    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
]
```

```python
PC1 = [
    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
]

PC2 = [
    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
]

SHIFT_SCHEDULE = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]

def hex_to_bin(hex_str: str, num_bits: int) -> str:
    """Convert hexadecimal string to binary string with padding."""
    return bin(int(hex_str, 16))[2:].zfill(num_bits)

def bin_to_hex(bin_str: str) -> str:
    """Convert binary string to hexadecimal string."""
    return hex(int(bin_str, 2))[2:].upper().zfill(16)

def permute(block: str, permutation_table: List[int]) -> str:
    """Permute the input block using the specified permutation table."""
    return ''.join([block[i-1] for i in permutation_table])

def left_shift(key: str, shifts: int) -> str:
    """Perform left circular shift on the key."""
    return key[shifts:] + key[:shifts]
```

```python
def xor(a: str, b: str) -> str:
    """Perform XOR operation between two binary strings."""
    return ''.join(str(int(x) ^ int(y)) for x, y in zip(a, b))

def generate_subkeys(key: str) -> List[str]:
    """Generate the 16 subkeys for each round of DES."""

    key_bin = hex_to_bin(key, 64)
    key_pc1 = permute(key_bin, PC1)

    left = key_pc1[:28]
    right = key_pc1[28:]

    subkeys = []
    for shift in SHIFT_SCHEDULE:

        left = left_shift(left, shift)
        right = left_shift(right, shift)

        combined = left + right
        subkey = permute(combined, PC2)
        subkeys.append(subkey)

    return subkeys

def f_function(right: str, subkey: str) -> str:
    """The Feistel (F) function used in each round of DES."""

    expanded = permute(right, E)

    xored = xor(expanded, subkey)

    s_box_output = ''
    for i in range(8):

        block = xored[i*6 : (i+1)*6]
        row = int(block[0] + block[5], 2)
        col = int(block[1:5], 2)

        val = S_BOX[i][row][col]
        s_box_output += bin(val)[2:].zfill(4)
    return permute(s_box_output, P)

def des_encrypt_block(block: str, subkeys: List[str]) -> str:
    """Encrypt a single 64-bit block using DES."""

    block = permute(block, IP)

    left = block[:32]
    right = block[32:]

    for i in range(16):

        prev_left = left

        left = right

        f_result = f_function(right, subkeys[i])
        right = xor(prev_left, f_result)


    combined = right + left

    ciphertext = permute(combined, FP)

    return ciphertext

def des_encrypt(plaintext_hex: str, key_hex: str) -> str:
    """Encrypt plaintext using DES algorithm."""

    subkeys = generate_subkeys(key_hex)

    plaintext_bin = hex_to_bin(plaintext_hex, 64)

    ciphertext_bin = des_encrypt_block(plaintext_bin, subkeys)

    ciphertext_hex = bin_to_hex(ciphertext_bin)

    return ciphertext_hex

if __name__ == "__main__":
```

```
plaintext = "17012005"
key = "133457799BBCDFF1"

print(f"Plaintext: {plaintext}")
print(f"Key: {key}")

ciphertext = des_encrypt(plaintext, key)
print(f"Ciphertext: {ciphertext}")
```

OUPTUT:



```
Abdurrahman Qureshi@DESKTOP-H2RV5MQ MINGW64 /d/Degree/SEM 5/CNS/Experiments/EXP6 (master)
$ py des.py
Plaintext: 17012005
Key: 133457799BBCDFF1
Ciphertext: A8FEDC190A80A64B
```

Performance (7M)	Journal (3M)	Lab Ethics (2M)	Attendance (3M)	Total (15M)	Faculty Signature