FCS Question 1

The AES algorithm is used to encrypt the plaintext: Two One Nine Two.
The key used is ABCDabcd12344321.

The following result is observed.
It is easily seen that the 1st state in encryption and 9th state in decryption match as expected and so do the 1st state in decryption and 9th state in decryption.

```
Test to Decrypt:b'Two One Nine Two'
The key used for AES:b'ABCDabcd12344321'

Initial_state:15352c642e0c06447f5b5d511467455e
1th state:a3df6cd69f3f1337c990bda9a84fe776
9th state:4de27cfc66def703455f53732ec13bd9
Ciphertext:b6c941e5ae3bcc5b3abfbbc6656128b2

Starting Decrypt
Decryption starting state:b6c941e5ae3bcc5b3abfbbc6656128b2
1th state:4de27cfc66def703455f53732ec13bd9
9th state:a3df6cd69f3f1337c990bda9a84fe776
Final state:15352c642e0c06447f5b5d511467455e
Original Plaintext:Two One Nine Two
```

There are 3 main components of the AES algorithm.

1. **Key Expansion**: This function expands a 128 bit and generates a key schedule. Since AES has 10 rounds there are 10 round keys. Each key has 4 words of 32 bits each. The round constants and the s-box used was taken from online sources cited below. Each word is transformed using a left word rotation, substitute bytes operation using an s-box,XOR with round constants and a XOR with the previous word.

```python
def keyExpansion(key):
    key=bytes2matrix(key)
    #round_keys=[]
    Nr=10
    Nk=4
    i=0
    while(len(key)<4*(Nr+1)):
        word =list(key[-1])
        if(i%4==0):
            word.append(word.pop(0))
            word=[s_box[b] for b in word]
            word[0]^=r_con[i]
            i+=1
        word=xor_bytes(word,key[-4])
        key.append(list(word))
    return key
```

2. **Encrypt:** The encrypt function is used to create ciphertext from the 128 bit plaintext. We create an initial state by adding the first key. Now we run 9 rounds with each round having operations-substitute bytes using an s-box,shift rows operation,mixing of columns,Addition of the round keys(XOR operation). In the 10th round mix columns operation is not performed. The final state is returned and converted to hex format which is the ciphertext for our plaintext.

```
state=bytes2matrix(plaintext)
expanded_key=keyExpansion(key)

state=addRoundKey(state,expanded_key[0:4])
print(f"Initial_state:{matrix2bytes(state).hex()}")

round_states = {}
for r in range(1,10):
    state=sub_bytes(state)
    state = shift_rows(state)
    state = mix_columns(state)
    state=addRoundKey(state,expanded_key[4*r:4*(r+1)])
    new_state=state
    round_states[r]=new_state
    if(r==9 or r==1):
        print(f"{r}th state:{matrix2bytes(new_state).hex()}")

state = sub_bytes(state)
state = shift_rows(state)
state = addRoundKey(state, expanded_key[10*4:11*4])


return state,round_states
```

3. **Decrypt:** The decrypt function uses the ciphertext produced from the encrypt function and key schedule from key Expansion to obtain back the original plaintext. It uses the same operations as encrypt but in the reverse direction. The first round involves an operation of adding the final round key,inverse shifting rows and substitute bytes using the s-box. Then 9 rounds are used with the operations-addRoundKey,inverse of mixing columns,inverse shifting rows,inverse substitute bytes. Final plaintext is obtained by adding the initial round key and converting the final state to bytes.

```
print("Starting Decrypt")

state = ciphertext
expanded_key = keyExpansion(key)
print(f"Decryption starting state:{matrix2bytes(state).hex()}")

state = addRoundKey(state,expanded_key[10*4:11*4])
state = inv_shift_rows(state)
state = inv_sub_bytes(state)
round_states = {}

for r in range(9, 0, -1):
    if(r==9 or r==1):
        print(f"{10-r}th state:{matrix2bytes(state).hex()}")
    state = addRoundKey(state, expanded_key[r*4:(r+1)*4])
    state = inv_mix_columns(state)
    state = inv_shift_rows(state)
    state = inv_sub_bytes(state)
    round_states[r] = state
print(f"Final state:{matrix2bytes(state).hex()}")
state = addRoundKey(state, expanded_key[0:4])
return state,round_states
```

Sources Used:
1)s-box/inverse-s-box and r-con taken from:https://github.com/boppreh/aes
2)https://onboardbase.com/blog/aes-encryption-decryption/
3)https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf