Assignment No 3

Title:

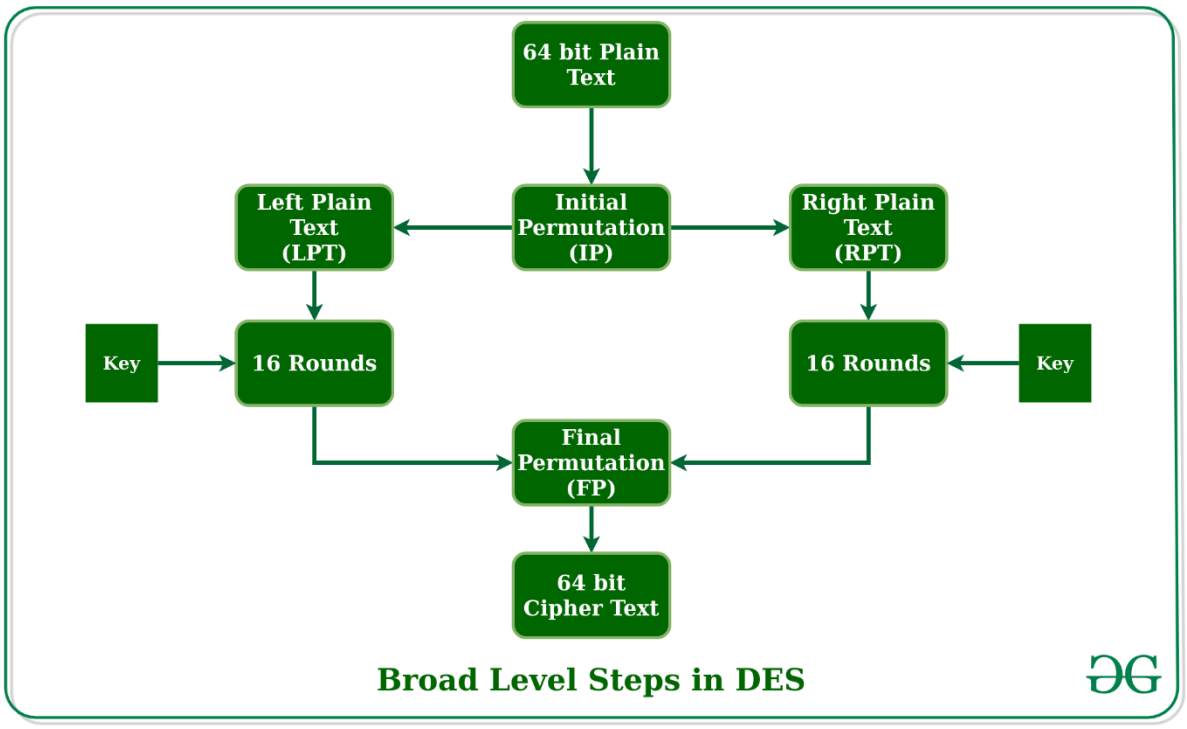
Write a c/c++/java/python program to implement DES algorithm.

Theory:

Data Encryption Standard (DES)

DES is a block cipher algorithm that takes plain text in blocks of 64 bits and converts them to ciphertext using keys of 48 bits. It is a symmetric key algorithm, which means that the same key is used for encrypting and decrypting data.

General Structure of DES is depicted in the following illustration –



Working of DES:

We have mentioned that DES uses a 56-bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.

Thus, the discarding of every 8th bit of the key produces a **56-bit key** from the original **64-bit key**. DES is based on the two fundamental attributes of cryptography: substitution (also called confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

- In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
- The initial permutation is performed on plain text.
- Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).

- Now each LPT and RPT go through 16 rounds of the encryption process.
- In the end, LPT and RPT are re-joined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64-bit ciphertext.

Implementing DES using Python:

Before we start, let me give you a brief introduction to the DES algorithm. DES stands for Data Encryption Standard, which is a symmetric-key algorithm used for encryption and decryption of data. It uses a 56-bit key to encrypt and decrypt data in blocks of 64 bits. The algorithm consists of 16 rounds of encryption, each round consisting of several operations such as permutation, substitution, and XOR.

Now, let's move on to the implementation of the DES algorithm using Python.

Step 1: Import necessary libraries

We will be using the `pycryptodome` library to implement the DES algorithm. You can install it using the following command:

```
pip install pycryptodome
```

After installing the library, we can import it in our Python program as follows:

```
from Crypto.Cipher import DES
```

Step 2: Define the key and plaintext

We need to define the key and plaintext that we want to encrypt. The key should be a 56-bit binary string, and the plaintext should be a 64-bit binary string. For simplicity, we can define them as follows:

```
key = b'01234567'
plaintext = b'abcdefgh'
```

Step 3: Pad the plaintext

The DES algorithm requires the plaintext to be a multiple of 64 bits. If the plaintext is not a multiple of 64 bits, we need to pad it with zeros. We can use the following function to pad the plaintext:

```
def pad(text):
    while len(text) % 8 != 0:
        text += b'\0'
    return text
```

We can call this function to pad the plaintext as follows:

```
plaintext = pad(plaintext)
```

Step 4: Create a DES object

We can create a DES object using the key as follows:

```
des = DES.new(key, DES.MODE_ECB)
```

Step 5: Encrypt the plaintext

We can encrypt the plaintext using the `encrypt` method of the DES object as follows:

```
ciphertext = des.encrypt(plaintext)
```

Step 6: Decrypt the ciphertext

We can decrypt the ciphertext using the `decrypt` method of the DES object as follows:

```
decrypted_text = des.decrypt(ciphertext)
```

Step 7: Remove the padding

After decrypting the ciphertext, we need to remove the padding from the decrypted text. We can use the following function to remove the padding:

```
def unpad(text):
   while text[-1] == 0:
      text = text[:-1]
   return text
```

We can call this function to remove the padding from the decrypted text as follows:

```
decrypted_text = unpad(decrypted_text)
```

Step 8: Print the results

Finally, we can print the plaintext, ciphertext, and decrypted text as follows:

```
print("Plaintext: ", plaintext)
print("Ciphertext: ", ciphertext)
print("Decrypted text: ", decrypted_text)
```

The complete Python program to implement the DES algorithm is as follows:

```
from Crypto.Cipher import DES

def pad(text):
   n = len(text) % 8
   print(b"text to encrypt:"+text + (b' ' * n))
   return text + (b' ' * n)


key = b'hello123'
text1 = b'Python is the Best Language!'

des = DES.new(key, DES.MODE_ECB)

padded_text = pad(text1)
encrypted_text = des.encrypt(padded_text)

print(encrypted_text)
print(des.decrypt(encrypted_text))
```

Conclusion:

The information security can be easily achieved by using Cryptography technique. DES is now considered to be insecure for some applications like banking system.
By adding additional key, modified S-Box design, modifies function implementation and replacing the old XOR by a new operation to give more robustness to DES algorithm and make it stronger against any kind of intruding.