Problem 1:

- Code explanation

All the permutation, s boxes, as well as functions of generating the round keys, substitute and get encryption key are provided by the Professor in lecture 3's code, so based on that, I first do the encrypt. I turn the input file into bit-vector and create a empty bit vector to store the output as the progress goes, as I think handling bit is easier during the progress. I then read 64 bit at a time for the input file bit-vector as one block, if the block is not 64-bit long then I padding 0 from its right side to make it 64-bit. And then as the DES states, I device the block into two – LE and RE, and follow the Feistel Structure states in the lecture's note as well as the hint by the professor(xoring some bit-vector), finally the output bit-vector is the combination of the latest LE and RE(LE+RE); as required by the problem, I then transfer the output bit-vector into hex string and store it into the txt file. The decrypt is bscally the same as the encrypt, the only two difference is I manually count the number of blocks since the input file is now in hex string format, and the round key need to be reversed since it's going backward.

-encrypted.txt

c6d80665196641a1bba15e161d81395b152197d2dcade2d203105c559383da6a81892f
0cfba8f349aaab0edfbbc420a47af2e1fce559a2163a520e9fc00211e2f73ea73d511cd1e1
7dc11052e31ed2996e755b2af37e79abcdb0f0989680d1ccde13f9858462d0575139cdd
53c61d0335132ad498d7b81c1cb8217b731760c78104d829cbf0ece166b0fe2d15cc49d
c863716fdafab205e3642c3cbfbdc1339c9ef905b38e2e36b2c4dfda1bc00d31c0ebea00
311a47c6e54d7642f8d3de4396fe1ba0ee3309585713612c1351a8546bb0b18dd66d2de
e64554d22046309e59699975d414e1d4dd2ffc1d052af0f5ae10392195cf8351a6d2301
0e07750ef4e1c6ed16d52fafa8d632b1cdd20a4051ec5961a50f8ecbdaeefce77b9c83effe
16af00c301fab16eef790a756a3cb6b97fdcb49467fa265a119ba81792231ae15ab97140
aad0e1e2e2843feaea9723793f4ae46ee68884ce855e0de291f63f723e32058fb537b994
35db0035e0ea8b0a44516d3c07860c11f31ce612331b1d32dd0df5aef5c75f59a8a16b3e
743081f4da8a70a3b7540061ca8a741b39d1d54f26aa9b205d59f1e6f72a2a849ea3c3af
9f0434f8ecac2aa556b0a77208a50eed8fd7df099e5956c3bc88137c28a3ebcf4baa189b6
16987314484643cd7f23bb9d314d4495f096d918131c6125ea36ecea4d24fd56b1caf92
7fb9e2ac224bca4d1907e54ef31c4f5e39bb9c6bc3b60bd6dda0d2c1e1983d819525859
09b01ed9020148ef776977cde0898495cdefd509fc654dd5b87c994aa8e122b12caaa80f
ccb6c9f15656275749ba1bfe006aeab6993c92cd409c7fb28f6feb345590730a0ac97087
2ad95895f69673b328760b4c1e1d58e28539e117d8f288e18c5da7eb84e7db47f68ddc8
7bcc02aeb27e15ae90ed20ecfe9c827073e6d7c56e15cef4d3565f15a7cba6a36f8e7d9e1
ee51ddebd3d0a8bc5a7b3fdc8e4aa4113c6ebe6c7bc0c2f1b2afe5159efbfd9b6e0312442
2dcb9c476a1f1ab40c796e12af3066982330065b18b7e1149b7144db2f03b4c876f5139
370752d121eb78e6f20c409b0e92e555cf6d87907dd50222e5531a37c58dcc61c630cbf
cc0d53f61bddce64f2178fabdf7b02f279d3f1d8d2103c70909ec5fe143cc9aaeef07fe323
cb803646cc5c687bbe5d1eb071a72ce6d92601e80c2e7290bb1313ceaa93c2af42fd8b4f
430aa371eb123e490bcff9e18d4407db1607f3b655e02b0883556d40f4edf3d63439f91d
e6112e388f534be43e6c540178f58d54ebd0be617235e6c8d2d9e0850a46f51e1689939
73532f7e97d2990063788531d0c0831bcb232f7e97d29900637cefc0c7b39ebd4dc32f7

e97d29900637fc33e6e02f4bafd532f7e97d299006374bca1971770d1a6632f7e97d2990
06371bb599c88aec1d163844e9837a4ba27c4e805bd8017d74050ec365b6e9192729ec
d9443195fadea4dfeb06aefe0c2cb3b8fc0ce0e806d6df42d85a4cef1c2852fcd77dc415d
494382cd8dfbd854b2578c8f0970e8bc5e57264a3ef16b730afd81235ae66053429bb93
23e912d22b7d06e19c4b3c9259319d

-decrypted.txt
Smartphone devices from the likes of Google, LG, OnePlus, Samsung and Xiaomi are in danger of compromise by cyber criminals after 400 vulnerable code sections were uncovered on Qualcomm's Snapdragon digital signal processor (DSP) chip, which runs on over 40% of the global Android estate. The vulnerabilities were uncovered by Check Point, which said that to exploit the vulnerabilities, a malicious actor would merely need to convince their target to install a simple, benign application with no permissions at all.The vulnerabilities leave affected smartphones at risk of being taken over and used to spy on and track their users, having malware and other malicious code installed and hidden, and even being bricked outright, said Yaniv Balmas, Check Point's head of cyber research. Although they have been responsibly disclosed to Qualcomm, which has acknowledged them, informed the relevant suppliers and issued a number of alerts - CVE-2020-11201, CVE-2020-11202, CVE-2020-11206, CVE-2020-11207, CVE-2020-11208 and CVE-2020-11209 - Balmas warned that the sheer scale of the problem could take months or even years to fix.

Problem 2:
- Code explanation
The encrypt process is almost the same as problem one. But before get into the encrypt process, I first separated the header and data, I do this by reading the file in binary format and separate the first three line and saved them into bit-vector format, the remaining content of the file is the data and also need to be stored into bit-vector format. Then the encrypt process is basically the same as problem 1, so I reused the code from it, the only difference is due to the missing of header, and the format of the input file, I manually count the number of blocks during the encrypt process, just like what I did for the decrypt at problem 1. Finally, putting the header bit-vector back to the encrypted data bit-vector, and saved it in binary mode.
-image_enc.ppm