Daniel O. Furry
ECE 40400
Homework 2

Problem 1:

My DES_text.py file begins by importing the key provided and making it a bit vector and then permuting it. Next, the round keys are generated by a for loop that adds each round to a list. Depending on which command is used (-e or -d) the program imports either the message file or the encrypted file text and converts it to a bit vector to be passed to my encrypt function. It is called through a for loop that iterates 16 times for the correct DES structure. The encrypt function takes in either the message or encrypted text, the round key, and iteration number. The first part of the encrypt function is some parameter checks to make sure the bits are read correctly and padding. Then it takes 64 bits at a time and permutes the right side then XORs it with the round key, then substitutes with the s tables and permutes with the p-box table. Finally, it combines the original right side with the modified right side. This process happens until the file has been read completely. The encrypt function returns a bit vector of the full encrypted or decrypted text. Lastly, that bit vector is converted into ascii values and written to the 4th argument in the call line.

encrypted.txt:

d04a94419ec6556c20029c83a277790c5c6380595291ecc23a40b90d60ae5b114dcefad2a37652e8
0dbed6bea5ab59d92b8f043c65e1ced023bfe2aa6c4e162de19db8a75ff0f779baa3629395da7d740
e784fd3150db010f0054cd9f70a13ad6a553f954a79ca990dadc1a697ed8821548099cadedb2d6572
810b06df68150cd16af08948628fab087c8577826ee1e0ca728ea3def08044613e608e9ee27cf91a7
052f2d11e6a42b371c5216e296a5486887d331794502300e42cfe9b228da863420c7a9d2eb3797bf
08185451fc5948a61890e2fec008abe98af6a313ba886300a3041f4ca3f273f177fdd95fb97cfd7724
c196421848826c105892bfbbe47e64551e146fc6d7130d00a2dd01fa6b14a6fb6fb054f843710ddd9
a311d54882db94802ceac4fd454332747d76b4e6be9e614545db3e6a8517c413628268c07aa64f71
75ce8cd40a00a86fb279fed136146b2f863a0f54bb9407a21418641ba55b6641e1acb69bdf816a2cf
41479f80ddde5a4a43da9f53758f152f58bb5919b65d4a80250c259b38f498f354223cbbcbe14e540
8aadc581eb0d5b19ef8219fbe42edc4e9f0467826a5c1a8141af67f0a897b4f212e5ab49417b576aba
488381be68fc72080ee3ed00b56152e2d7da477b92c98379b694d4f466eb0d93d083fd62d36ef1ca
7f3b4399af80559ffbe0bd48b6eb441a569d479f94a54cb9ca816990971e229831db528e70972cae2
f82df38026db9db5b118ff17df3a7621911b51626ab948dd95a777b4219b0ad0ab6180def71f24b42
b23444d03b974681d583e07040d443d9365241e1fa77e1b4684da92913a6ea9a2af407d586ddf8b2
42706e8775ced9fa520291bbafe441dfab3c4b5d93cfe1654202d0b7ff5c381a6a2c489e2c756eb40
b6b98482a49878d04f4422fcf43605826dd6dc32cd8679e51bc800e3ae48673c19c5890c7eec8fc58
775299ea756be20afff89395ac6b021f5bb37c36e30f5948979c96b76537d8785721f1b9789e325d2
e779c4e0859c093ba756c8998219cfc497f0b7f66e259eebea3fba7a9ceed545ed833506d558c2dd9
a8812ed9bdc69e9b0bdfbd514399d2a43be6bf50a2ddab68b3c3b449a430efdd46755871a8697737
a7fd251de37390186a0c701ef7839a2b2ee99a8d6aaf540aefd111c8507120fbc296ade7e0a30846b
4ad461f2af4db654e8e0008fa5a2fa42381d8350bd431d714c42f478ca43e3f31d4e2b77c4fea7b5fc
92b55c18fd29ac06b78797333758a54e7aab3439dc079d168b7c416e23cd49084a57ff1c0974dd36
102983521b30ecd7fd1931201daab5059b8139f5a3017cd7fd1931201daab744fae27721dad95cd7f
d1931201daababcce6cc5c4ddacecd7fd1931201daab462cde434ec9b646cd7fd1931201daab0013d

a3321192b13b1bcd34158bc5878e3dbd126d0b7edf4cb345a27fa36e8df45ed30ec1b4cf954fd1fb2
eb165e3fde33aab34a81ef30b95855c1917ea1826f0093dfae7a9e6124ac8036677dc75ddb8cc7954
8b5f6b673e2aaa2e74de559d17d4c3597eb793828ce2eba373130b87fd5684085959a1496

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:

My DES_image.py file first import the key and generates the round keys identically from problem 1. Next, it imports the ppm file by reading starting at line 3 until the end to be passed into the encrypt function. It also reads the first 3 lines and saves it as the header and converts it to a bit vector using a for loop for each line. The guts of the ppm file are also converted into a bit vector in the same way with its own for loop. Finally, the encrypt function is called 16 times with a for loop. The encrypt function for this problem is identical to what is described in problem 1 because the DES method of encryption doesn’t change. After the encrypt function is done cycling through, the newly encrypted text is combined with the bit vector header. This combined file is then written to the out file.