Homework Number: 2
Name: Dorien Penebacker
ECN Login: dpenebac
Due Date: 01/26/2023

My code uses many portions of the given code from the lecture 3 downloadable code. These are noted in the .py files and I will note them here as well.

Problem 1

DES_text.py :
Starting with my main function, I use the argparse library to parse the inputs. I use a argparse.ArguementParser() group to choose either encryption or decryption. The argparse module also parses two more inputs for the files. Depending on whether decrypt or encrypt is called, the filenames and locations are still the same. I used this documentation to debug my argparse code. https://docs.python.org/3/library/argparse.html.

DES_text.py encrypt :
My encrypt function first opens the key file and reads the text from it. Then it uses the get_encryption_key() and the generate_round_keys (both given), to create a list of bit vector keys used in the DES algorithm. It then uses the BitVector module to read into a bit vector from a file using the filename= format and uses the more_to_read() function in a while loop. It then reads 64 bits at a time. If there are not 64 bits from the file, it will use the pad_from_right() function to add the remaining bits by zero, extending it. The code then splits the bit vector using the divide_into_two() function then runs it through the feistel function 16 times. The feistel function takes an input of a left hand bit vector, right hand bitvector, and a single round key. This round key is taken from the list created earlier. The feistel function has been implemented based on figure 4 from the lecture slides and using the p-boxes from the lecture slides. The expansion permutation function, xor function, substitution function, are all given. The permutation function used is just a permutation function call with the p-boxes taken from the lecture slides. It then returns the appropriate left and right hand for further calls. It then appends to an output bit vector. After the encryption finishes, the function writes the output to the file specified.

DES_text.py decrypt :
The decrypt function follows the exact same process with small differences. The first difference is that it reads from the file 128 bits rather than 64 bits at a time since the text is stored as hex string rather than regular strings. The second is that when the output bit vector is being appended too, it appends the right hand first rather than the left because of the reverse methodology of the decryption algorithm.

Encrypted:
36d2e582921b6b4a4729ec8a60a4915ba76f3fec1c010014c13444b4afbefb124743582e779a57c
f992d871fcd7e178fe0c5b2c8ccc1a78fcae1aab4c09dd92388d20af1deaf36212e9fad48d6cf32d8

```
299cf7bfe82e8faa32b3383d1877fb86eb489571936cdcda5d32f1bc9a359bd63f411305859fec91
2107c147cb77b2f459f944561933e2ca54416929a35c2ce30438568de299dac4a33811a43d6b1e
6ec75f86e0768b8ff5eea71a6bb8907125a17a19997c153b4665123bf24bfe084f129a72292fe22f
adf0ab59a06babc93f9aecc82545e35920fa68a6eea18322458bf5a0fe9e50695326cb0ff211484b
883a677b20a3318584f058b818fa594e9bb2744c67a5ba2ad2d65e39d4522476efa8770e1bf554
7cc90f12f73ec93102586e55c8a8e6bdeb8e16205040647bbcb8be20b29d589da8c3fa2a9ec2f00
dc056046c299bbb1532ef8c38b24c021558175055c4a95a1b193deec41112afa5db015fbac30c6c
95c83e3cb07f9b28c849b0330d4b4e84abf996f91ae58a499a44b87340c11ca00748b00072d7bf2
2bb383f3f2e2aa185921e974e23fc695bab5c2ddd27d5fa0e6e6de2af262f2608fa8cbc25bfbdc4f5f
8f0f785a1b4d4c63fa94f0c16601d8cff74856ca0a1ca8e1167db0a5a55e7dbb246202ae59835c16
e90c1e0c5b2c8ccc1a78f726e8963d971baba5db79b6739f3fa4329acdfef24b1b13d361832c5bd
814d7acf7059e1b251f74e604116ecb90755cc43a12639c01917653cd945c9065737efa9401947f
b9557568b567bdf059a474f95217f55ba63b3ed666854c2dda688b6acf0722076e3fd18d59b9109
d4639c5a10dcc9dd17a3e78fe956fb9687276ad8aefbfa2764ab669e7444e751fc396940fee2446
b2e40d29f277a46ab9781445b25725cd74215a01694f2566b33456851c5966303a2053f6a22d41
581fa810f1668eb7761db9206b466a8a65e50171f030c680a971cffd17e583060cd6e32ec5bd4ba
1f9bda5976a883327bada116974b7e8220290949d5315cd4d308e297b7789bcf7466c433e6effef
150ea4a44df492f449509044104c47b32351b272672fc599ea6926482920a08dd08cfdfdd19ae50
585efebe84f51afbd7487e04b5e127457e37e615da2b55fafc317fecebf59a
```

Decrypted:
In the unforgiving world of Formula One, Lewis Hamilton abides at the top. He's the man to beat, the top earner, the most important voice, the most prominent figure - a Black man alone at the summit of motorsports' highest echelon. England's knight in Mercedes armor. Over the past 15 years, the 36-year-old Briton has won seven world championships, tying the record set by Ferrari's Michael Schumacher - the German F1 driver who was regarded as the greatest of all time until Hamilton broadsided him from that perch. At Sunday's Russian Grand Prix, Hamilton rallied through a late rain shower to claim the checkered flag on the way to becoming the first driver in the sport's history with 100 career victories. And that's besides his 100 career pole positions. As achievements go in racing, this is beyond otherworldly.

NOTE:
I’m writing this in google docs so it cannot copy the decrypted text absolutely. At the end of the sentence; “otherworldly.”, there are null byte characters that cannot be seen. Because of this I will also be adding my decrypted.txt and encrypted.txt to the submission. Using hexdump, my last line is : 726c 646c 792e 0000 which correspond to “rldly. “ with the two extra spaces.

Problem 2

DES_image.py :
The DES_image.py uses the same main functionality from DES_text. The main difference is that it first reads out bits from the file until it gets 3 newlines. This parses out the header from the file input and allows the encryption to encrypt the rest of the file. The output also rewrites the header back into it.

[illegible]