Luke Canfield ECN Login: lcanfiel Due Date: 1/25/2024

HW02 ECE 40400

Problem 1 Encrypted Output

```
3d108f88993494ad6f176296433352894c553468c9abc401a425d05443b4a9f60364daecc4d850cded8599
0829b800a6c17ceb0c55e399664e95f49b25e186d42151d0701965dc2c493c43d2136a8c36939f7742224
52cf75cb8d76b0b27aa1c029a752faab4da8b4e59522bcde1170cba4850733556789463cc5fac970a1a8aef
74d9e8807085c072f1c4eede3b155f171b0246c88d112458d18c982a625cbd1cdea553feb7272775a7cd3c
703d9a3f693afa5cfac3ee0bde02b6811f9812edf12a237b1683b1e81537d9d9d819d151ce1be39f4015ba
5d7f75fad2f02fafeda1ca59cdbe75eddf4c553468c9abc401f94219c72878d76c1c3d1a10e3a520760b4d70
ac46eb32b73dd8a16cb7f98daf86ff8b394bd552f18b1fbf529a8da6d9267bee11c012400441e8419f051dd
13a2afeda5653645729b828ddece92132805bc08f30ffe2bc20956b49b1516bd5d8ce61e0809a5910945c4
a0b4575cc78d1fc5d711daac0603a7b50f66e5ef13b02590cd7c10c68326a2d83c23ae2db2b4f333e8a0ec1
3f3e9109a820602f911e656584800b596b5bdaf5ad0a816545e496f17a66f530d1f08d4dab64ef595f9963b
a8b90df8530a9565dca04314abfabc73d3cf518297e0cdc45bd78af69523ec6a35c60a137a91e14bf2e19a9
e3041a5d684f0f3f6e899e77e5650ad049d4ab4c807bdd4a1c91768d57a414b3eaef307af03068e59d47f0c
a906363b40d2096aa9d1bc2f2447350d6d8402b17f9ac0e06247ecd4da9bbba3d8b1fbf529a8da6d92728
4ac407140c92d543a77a115fa064cc10ac3ee4b81d222c077e80e8052dfec8af0eeda0d0d214fdaa978b097
283cf708ab6591abab296fe94f362b700fb1b932f56c8e86bde281adbc7312f86b127242202fb255829cf74
1074de5f9f324d58abd999e7f655a0056e9dd0b7604daecb17836d33cfce4747e8e0dc6caae7b799c92ac2
3d8c4ab5431bc4b03862494c0529414afa3e631400e1b8d0e307c4161d2d6ff7c6f79849fba3aadea4207f9
86e3a700623142549a485954d2a04de10df7d97d3870317df8bc37f59515fb02fa408970b2693949e1923d
184ed6b8bfb1e19476bd5af553268b5dc87b76ddf3520f13086e056c5e48eb5fe5e17b3f18603510095272
6958c6224bcd2ba4dcb1f93731bda0085ee016478e19fdbf3cc498b1a7f3ddf99383c28497d0155d8db5dd
c2a7efd759ffc178522d8d3648b1b37f45a2ef9115976fd08e0780cea1fef470a90cfe98d0be47e35cc5c94b7
a2736c27bb2f3e69b23150259d4f51e1dd86f240fd56ec388ee171b9f117b4c4b76e21a80db635b7ab4bfbf
827820e76db68177585e908ec10978dcc1c10f57ca8c9a5
```

Problem 1 Decrypted Output

Scuderia Ferrari is the racing division of luxury Italian auto manufacturer Ferrari and the racing team that competes in Formula One racing. The team is also known by the nickname "The Prancing Horse", in reference to their logo. It is the oldest surviving and most successful Formula One team, having competed in every world championship since the 1950 Formula One season. The team was founded by Enzo Ferrari, initially to race cars produced by Alfa Romeo. By 1947 Ferrari had begun building its own cars. Among its important achievements outside Formula One are winning the World Sportscar Championship, 24 Hours of Le Mans, 24 Hours of Spa, 24 Hours of Daytona, 12 Hours of Sebring, Bathurst 12 Hour, races for Grand tourer cars and racing on road courses of the Targa Florio, the Mille Miglia and the Carrera Panamericana. The team is also known for its passionate support base, known as the tifosi. The Italian Grand Prix at Monza is regarded as the team's home race.

Brief Explanation of Problem 1 Code

All of my code besides the encrypt, decrypt, and main functions were adapted from the professor’s lecture 3 code that was shown in lecture or given to us in the HW02 zip folder. My main function controls whether the encrypt or decrypt function is called within the DES class based off the first argument given in the command line. If “-e” is the first argument the encrypt function is called which continually iterates through the Feistel structure implementing each step (ex. Expansion permutation, substitution with the s-boxes, permutation with the p-box) until there are no more bits to read in the BitVector. It then converts the bits into a hex string and writes it to the encryption file. If “-d” is the first argument in the command line the decrypt function is called which iterates through the Feistel structure in the same way as the encrypt function except the round keys are reversed so that they decrypt the message. This file also converts the input encryption file from hex to binary so it can iterate through the steps of the Feistel structure. The output is written to the decrypted file and then the encryption file is written over with original hex input.

<u>Encrypted PPM Image</u>

Was never able to successfully display ppm image :(

<u>Brief Explanation of Code Problem 2</u>

I’m pretty sure my code doesn’t work but there’s a slim chance it does. I wasn’t able to view the file we’re initially given in GIMP or online PPM image viewers so I truly don’t know if my code is correct (I know it resembles helicopter). That being said my new function encrypt_image is very similar to the encrypt function. I read the first three lines of the input file and set that as a variable named header which is a byte string. I then open the output file and write the header to it before encrypting the image data. I used the Feistel structure as before to encrypt the data and then write it to the output file afterwards. My code would run for about 45sec but like I said I was never able to view the image. I’m assuming I messed up somewhere but it could be correct so rather be safe than sorry.