Question 2:

1. Let temp\_block denote a Sage variable that contains the output of the first application of function *fK* (f\_K in the Sage example code) while encrypting with Simplified DES. Using subroutines from the example sage Code, write sage code to recover the input block passed to Simplifed DES Decrypt. Meaning reverse the first steps in Simplified DES Encrypt. You may assume that you have the first round key in a variable K1.
2. Using subroutines from the Sage example code for Simplified DES, write a function to compute Simplified DES Decrypt.

Solution:

1. Let temp\_block denote a Sage variable that contains the output of the first application of function *fK* (f\_K in the Sage example code) while encrypting with Simplified DES. Using subroutines from the example sage Code, write sage code to recover the plaintext block passed to Simplifed DES Decrypt. Meaning reverse the first steps in Simplified DES Encrypt. You may assume that you have the first round key in a variable K1.

Using subroutines from the Sage example code for Simplified DES, write a function to compute Simplified DES Decrypt.

#

# Simplified DES Decrypt Function

#

def SDESDecrypt(plaintext\_block, K):

(K1, K2) = SDESKeySchedule(K);

temp\_block1 = IP(plaintext\_block);

temp\_block2 = f\_K(temp\_block1, K2);

temp\_block3 = SW(temp\_block2);

temp\_block4 = f\_K(temp\_block3, K1);

output\_block = IPinv(temp\_block4);

return output\_block;

temp\_block2 = f\_K(temp\_block, K1);

plaintext\_block = IPinv(temp\_block2);